Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRAO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN INGENIERÍA DEL SOFTWARE

Sistema de Gestión para Empresas de Venta de Productos en Eventos Comerciales

Estudante: Marcos Rial Troncoso
Dirección: Fernando Bellas Permuy

A Coruña, septiembre de 2020.

A mi pareja y compañeros, por el apoyo recibido durante este recorrido.

Agradecimientos

Quiero agradecer principalmente a mi pareja, Andrea, por su esfuerzo y apoyo que han hecho posible superar este reto. Gracias por la oportunidad y confianza puestas sobre mí.

También a aquellos amigos que han surgido durante la carrera, con los que he compartido esta experiencia y de los que he disfrutado rodearme, creando valiosos recuerdos.

Finalmente a mi tutor, Fernando, por sus consejos durante la realización de este proyecto y las charlas que hemos mantenido, que me han motivado a indagar más sobre esta profesión.

Resumen

Este proyecto tiene como objetivo el diseño e implementación de un sistema de gestión para empresas dedicadas a la venta en persona durante eventos comerciales. Con este sistema se pretende cubrir las necesidades esenciales de su proceso de negocio, como la gestión de un catálogo y el registro de ventas. El principal componente de la solución será un servicio REST empleando el framework Spring, el cual contendrá la lógica de negocio y se encargará de la persistencia de los datos. Se dispondrá de una aplicación web SPA para la gestión del catálogo, en la que se empleará la librería React. Por último, una aplicación de escritorio multiplataforma desarrollada con Electron actuará como punto de venta durante los eventos, trabajando principalmente sin conexión. Ambas aplicaciones consumirán el servicio REST.

Abstract

This project aims to design and implement a management system for companies dedicated to in-person sales during commercial events. This system seeks to cover the essential needs of the business process, such as the management of a store catalog and the sales record. The main component of the solution will be a REST service using the Spring framework, which will contain the business logic and will be responsible of the data persistence. A SPA web application will be available for catalog management, where the React library will be used. Finally, a cross-platform desktop application developed with Electron will act as a point of sale during events, working mainly offline. Both applications will consume the REST service.

Palabras clave:

- Gestor de comercio PYME
- Punto de Venta sin conexión
- Impresión de tickets
- Spring Boot
- Servicio REST
- Aplicación SPA
- Material-UI
- Electron
- PouchDB
- Scrum

Keywords:

- SME business manager
- Point of Sale offline
- Ticket printing
- Spring Boot
- REST Service
- Single Page Application
- Material-UI
- Electron
- PouchDB
- Scrum

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Durante todo el año se producen eventos de diferente índole a lo largo del territorio español. Sus organizadores solicitan un recinto donde tendrá lugar cada evento y se encargan de gestionar todo lo relacionado con estos, desde su promoción al público, las diferentes actividades que sucederán durante su ejecución, invitar a personalidades reconocidas, etc. En el caso de que estos eventos ganen popularidad, lo más habitual es que comiencen a producirse de forma anual, generando así una marca reconocible para su público.

A estos eventos también se suele invitar entidades o tiendas comerciales que tengan relación con la naturaleza del evento para que expongan su trabajo o productos. Algún ejemplo de esto serían los eventos de temática artística en los que se invita a ilustradores con cierta popularidad, eventos de exposición donde las tiendas presentarán las novedades de su catálogo, y muchos tipos más. Entre estos destacaremos los dedicados a la cultura japonesa, que tratan en especial el mundo del cómic y la animación de esta región, y que en los últimos años han incorporando en cierta proporción los ámbitos del cine y las series de actualidad.

En relación con los últimos eventos mencionados encontraremos una gran cantidad de pequeñas tiendas que se dedican exclusivamente a viajar participando en estos eventos, ofreciendo a los asistentes los productos de colección que expondrán en un estand asignado por los organizadores del evento. Estas tiendas suelen iniciar su andadura sin ningún sistema informático en el que poder agregar y consultar su catálogo de productos o en que realizar ventas, perdiendo valiosa información de mercado que podrían analizar. Una de las complejidades añadida para este ámbito es la existente proliferación de series, películas, series de animación y cómics; lo que genera un esfuerzo constante para estar al corriente de las nuevas tendencias e incluso aquellas que puedan resurgir, con el objetivo de mejorar la captación de clientela.

En esta situación se encontraba el cliente del proyecto, **Frikiland**, una tienda de venta de *merchandising* que quiere mejorar su proceso de negocio incorporando un sistema que le permita registrar las ventas que realiza durante los eventos. Una de las dificultades que nos propone es el registro de ventas sin conexión, dado que los eventos suelen producirse en recintos cerrados, con una cobertura poco fiable o directamente sin acceso a internet.

1.1 Objetivos

El objetivo de este proyecto será la creación de un sistema orientado a empresas pequeñas que siguen el modelo de negocio expuesto previamente, ofreciendo como funcionalidad principal:

- **Gestión del catálogo de la tienda.**
- **Gestión de usuarios con acceso al sistema.**
- **Registro de ventas** sin conexión e impresión de los resguardos de venta.
- **Visualización sobre ventas realizadas y resúmenes** sobre las mismas.

Dado que se enfoca en una empresa pequeña se debe tener en cuenta que los empleados podrán encargase de más de una tarea dentro del proceso de negocio. Debido a esta característica se hará frente a un diseño en el que un usuario podrá contar con múltiples roles.

Con la realización de este proyecto se pretende demostrar las competencias adquiridas durante el grado, haciendo frente a un cliente con unas necesidades reales. Además se emplearan diferentes tecnologías que no han sido abordadas previamente, tratando de incorporar así nuevos conocimientos durante el desarrollo.

1.2 Vista global del sistema

ShiftShop es el nombre otorgado al sistema ideado para dar solución a la problemática expuesta y está compuesto por tres componentes bien diferenciados.

Su componente central será un servicio REST con la lógica de negocio principal y que ofrece el punto de acceso a través del cual será consumido por el resto de componentes. En la implementación se empleará el *framework* Spring, el cual agilizará considerablemente su desarrollo, mientras que los datos serán almacenados a través del gestor de base de datos relacional MySQL.

El segundo componente será una aplicación web SPA (Single-Page Application). Su objetivo será otorgar una interface que permitirá a los usuarios ejecutar la funcionalidad expuesta por el servicio de forma intuitiva y cómoda en diferentes tipos de dispositivos. A través de esta se podrá gestionar el catálogo, los usuarios y visualizar las ventas que han sido registradas a lo largo de los eventos. Su implementación se basa en el desarrollo basado en componentes, logrado a través de la librería React.

Por último se cuenta con un Punto de Venta (PoS - Point of Sale), una aplicación de escritorio multiplataforma en la que se podrán registrar las ventas e imprimir sus respectivos tickets durante los eventos, todo ello sin una conexión a internet. Una vez de vuelta en la oficina de la empresa, los terminales PoS sincronizarán estas ventas con el servicio REST para su persistencia. En su implementación se hará uso del *framework* Electron, mientras que en el almacenamiento de datos se utilizará PouchDB, una base de datos documental embebida. Para la impresión de tickets se contará con una impresora térmica Epson TM-T20III, utilizando el SDK (Software Development Kit) Javascript que ofrecen desde su página para desarrolladores.

En la figura 1.1 se muestra la arquitectura del sistema y puede observarse los dos flujos de uso por parte de los usuarios. Para el flujo principal (A) los usuarios se descargarán en sus dispositivos la aplicación SPA que se hospeda en un servidor de contenido estático NGINX (A.1), y a través de esta podrán consumir la funcionalidad expuesta para su respectivo rol en el servicio REST (A.2), todo ello en la oficina de la empresa. Por otra parte el flujo para el registro de ventas (B) se dividirá en tres etapas. La primera de estas ocurre también en la oficina y consiste en la sincronización de los terminales con la aplicación PoS (B.1), obteniendo los últimos datos actualizados en el servicio REST. Una vez en los eventos se inicia la segunda etapa, en la que se produce el registro local de las ventas en los terminales por parte del personal de venta (B.2). Finalmente, de vuelta en la oficina, se procede con el registro de las ventas en el servicio REST (B.3), manteniendo así de forma centralizada estos datos.

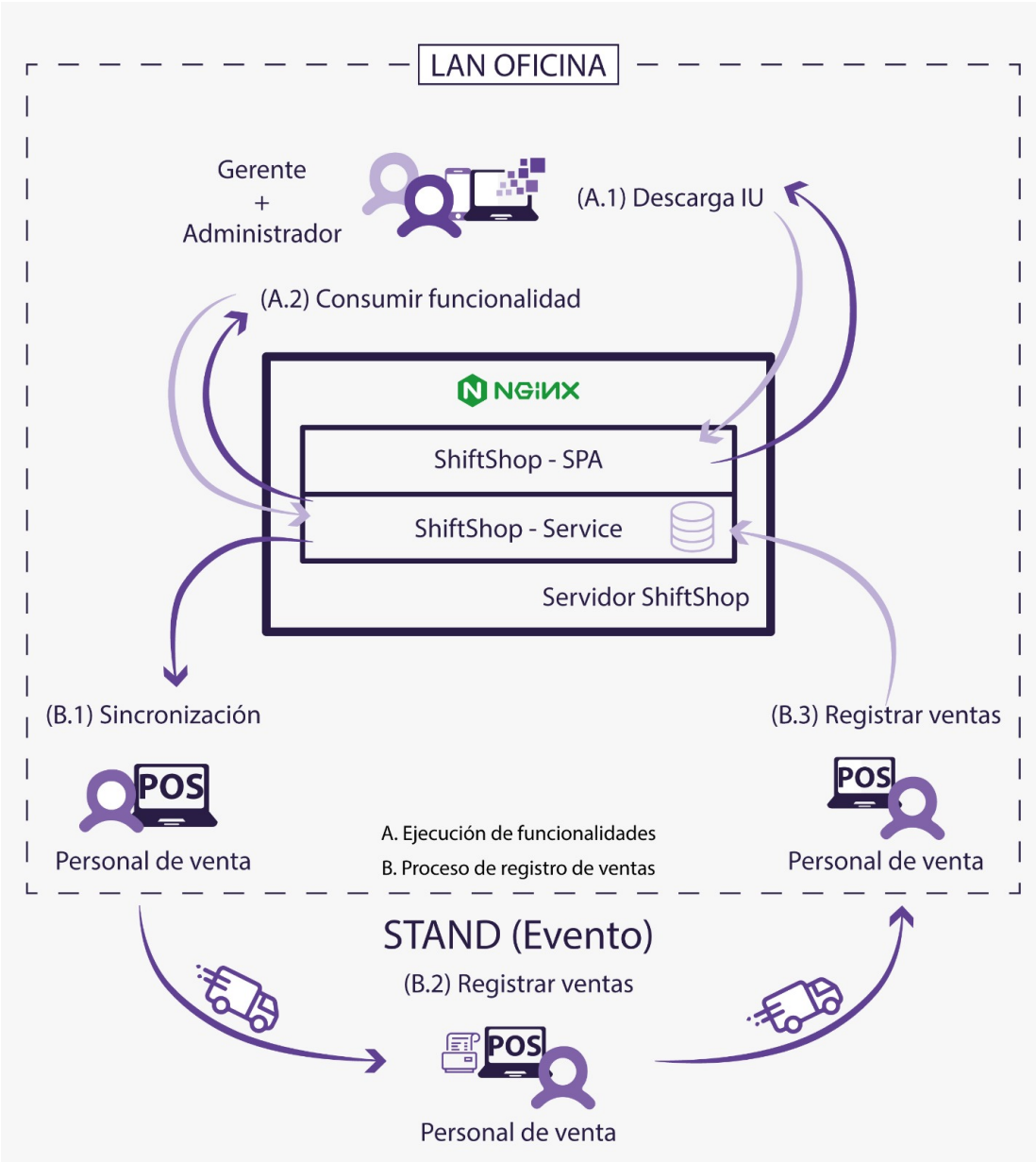


Figura 1.1: Infografía del sistema

Capítulo 2

Estado del arte

Actualmente existen diferentes herramientas que engloban la funcionalidad que se pretende cubrir en este proyecto. A continuación se analizarán dos de las soluciones más conocidas y empleadas, pero que cuentan con ciertos inconvenientes a la hora de poder ser incorporadas por las empresas con el modelo de negocio expuesto en el capítulo 1.

2.1 Shopify

Shopify [1] es una plataforma principalmente conocida por su solución para tiendas e-commerce [2], con un gran abanico de características entre las que podemos destacar: creación de sitios web a través de plantillas, compras online, gestión de clientes, gestión de pedidos y reembolsos, gestión de inventario, generación de informes, marketing y SEO.

Además de su solución e-commerce también ofrece una solución punto de venta [3], que combinada con la anterior permite a una tienda unificar las ventas que realiza, tanto físicas como online. Para esta solución se dispone de un hardware específico que deberá solicitarse a mayores y que se puede integrar con otros componentes adicionales como: impresora de tickets, lector de códigos de barras y lector de tarjetas.

Su solución de punto de venta junto con el gestor de catálogo e-commerce es lo que más se podría acercar al objetivo funcional de este proyecto. Los principales inconvenientes que se ven en esta plataforma son:

- Mensualidades: El precio por el plan más acorde a estas empresas es de 80€/mes, además de una serie de tarifas por realizar pagos con el sistema.
- Equipo: Para poder usar el punto de venta es necesario adquirir un equipo con un coste que ronda los 700€ (terminal iPad e impresora de tickets).
- Límite de usuarios: El sistema tiene un número limitado de tan solo 5 usuarios para el plan mencionado.

- Conexión a internet: Para poder acceder y utilizar el punto de venta es necesario tener acceso a internet.
- Roles para empleados: Para contar con esta característica es necesario agregar un plan extra con un coste mensual de 90€, pero que permitiría un número ilimitado de usuarios.

2.2 Square

Square [4] ofrece unas características muy similares a Shopify, pero centrándose más en su solución para el punto de venta. Una de sus principales ventajas es la posibilidad de utilizar este punto de venta sin conexión, una característica que aporta mucho valor en empresas de este ámbito. Entre otras características destacamos sus planes asequibles, ya que ofrecen desde un plan gratuito, con la funcionalidad básica para trabajar en un punto de venta, hasta tarifas mensuales (entre 12 y 26 €/mes) que permiten la incorporación de una solución de comercio electrónico con la que complementar la anterior.

Al igual que ocurre con Shopify, es necesario adquirir el equipo necesario para funcionar como punto de venta (su terminal también será un iPad) y con unos precios similares a los vistos. La necesidad de un terminal específico, junto con el elevado precio de su impresora de tickets, son el principal inconveniente al plantearse esta solución. Otra de las desventajas destacadas que se observó durante la prueba de este sistema fue la funcionalidad de búsqueda en el punto de venta, ya que a la hora de encontrar un producto se utiliza una correspondencia literal con su nombre, lo que entorpecería la dinámica de las tiendas objetivo ya que suelen tener un catálogo de productos bastante elevado.

2.3 Otros

Entre muchas otras soluciones se hace mención especial a las que ofrecen **Ecwid** [5] y **Vend** [6], las cuales comparten en gran medida muchas de las características descritas en las soluciones previamente analizadas, además de ofrecer otras únicas que las diferencian de sus competidoras, como la posibilidad que ofrece Ecwid para incorporar el catálogo de la tienda en otras plataformas de venta como Amazon o Ebay.

Capítulo 3

Análisis de requisitos global

A través de este capítulo se pretende describir los actores identificados que harán uso del sistema y los requisitos funcionales que han sido recopilados a través de las historias de usuario (capítulo 5) identificadas.

3.1 Roles

Analizando el entorno que se pretende abordar en el proyecto se pueden diferenciar tres actores principales para estas pequeñas tiendas, descritos a continuación:

- **Administrador**: Este actor está ligado a las tareas administrativas de la empresa, relacionadas con la búsqueda de eventos, búsqueda de productos y su incorporación al catálogo, seguimiento de ventas, etc. Tienen el puesto de mayor responsabilidad en la empresa ya que se encargan de su total gestión. Dadas las responsabilidades mencionadas serán los encargados de la gestión del catálogo del sistema y podrán visualizar todas las ventas que se han registrado.
- **Personal de venta**: Por otro lado nos encontramos a los empleados que se contratan como personal de venta en los eventos, cuya tarea se centra únicamente en el proceso de venta y el registro de las mismas. Estos pueden rotar mucho e incluso contratarse esporádicamente para eventos concretos, por lo que la información a la que tendrán acceso será más limitada.
- **Gerente**: Finalmente se agregó un actor más debido a la necesidad de incorporar un rol destinado a la gestión de los usuarios del sistema, otorgado en exclusiva a la persona de mayor autoridad dentro de la empresa.

Puesto que el número de empleados de esta empresa puede ser muy limitado lo normal es que uno de ellos pueda ejercer varios roles, por lo que un requisito del sistema será que se pueda asignar varios roles a los usuarios.

3.2 Historias de usuario

En esta sección se recogen las historias de usuario que en algún momento pertenecieron a la pila del producto del proyecto, en las que se expone una necesidad y el rol que la reclama.

3.2.1 Pila del producto

ID	Descripción
SHSH-5	COMO Administrador QUIERO agregar una nueva categoría al catálogo
SHSH-6	COMO Usuario autenticado QUIERO ver las categorías registradas
SHSH-7	COMO Administrador QUIERO modificar los datos de una categoría existente
SHSH-8	COMO Administrador QUIERO añadir un nuevo producto al catálogo
SHSH-9	COMO Usuario autenticado QUIERO ver los detalles de un producto concreto
SHSH-10	COMO Usuario autenticado QUIERO realizar búsquedas personalizadas sobre el catálogo de productos
SHSH-11	COMO Administrador QUIERO modificar los datos de un producto existente
SHSH-12	COMO Administrador QUIERO deshabilitar un producto del catálogo y poder volver a activarlo
SHSH-13	COMO Gerente/Administrador QUIERO realizar búsquedas sobre las ventas realizadas
SHSH-14	COMO Gerente/Administrador QUIERO visualizar resúmenes de ventas
SHSH-15	COMO Gerente QUIERO registrar nuevo personal con acceso al sistema
SHSH-16	COMO Gerente QUIERO ver el personal que hay registrado en el sistema
SHSH-17	COMO Gerente QUIERO modificar los datos de un empleado
SHSH-18	COMO Vendedor QUIERO registrar la venta de productos a clientes en el PoS
SHSH-19	COMO Vendedor QUIERO realizar la impresión del ticket una de venta
SHSH-20	COMO Vendedor QUIERO sincronizar los datos necesarios a utilizar por el PoS y guardarlos localmente
SHSH-21	COMO Vendedor QUIERO autenticarme en el PoS usando los usuarios locales
SHSH-22	COMO Vendedor QUIERO realizar búsquedas sobre las ventas locales registradas en el PoS
SHSH-23	COMO Vendedor QUIERO sincronizar las ventas realizadas durante los eventos en los registros de la empresa
SHSH-27	COMO Gerente/Administrador QUIERO ver los detalles de una venta concreta
SHSH-24	COMO Usuario no autenticado QUIERO autenticarme en el sistema
SHSH-30	COMO Usuario autenticado QUIERO cambiar mis credenciales para entrar en el sistema

Tabla 3.1: Historias de usuario en la pila del producto

3.2.2 Funcionalidad

En este apartado entraremos más en el detalle de cada historia de usuario, comentando aspectos relevantes sobre la funcionalidad esperada y que tendrán repercusión en la implementación de cada una de estas.

SHSH-5 Agregar categoría

Los administradores podrán dar de alta nuevas categorías. Estas deben tener un nombre que las identifique unívocamente, sin distinguir entre mayúsculas y minúsculas. Se notificará al usuario con un error en caso de intentar registrar un nombre duplicado.

SHSH-6 Obtener categorías

Cualquier usuario del sistema podrá visualizar las categorías que han sido dadas de alta. Se mostrará el conjunto total ya que no se espera un alto número de categorías y, en caso de existir ninguna, se mostrará un mensaje al respecto.

SHSH-7 Modificar categoría

Desde la vista de categorías cualquier administrador podrá seleccionar una de ellas para la modificación de sus datos. Al igual que ocurre al agregar se debe tener en cuenta las mismas restricciones sobre el nombre.

En la figura 3.1 se muestra el mockup de esta vista destinada a la gestión de categorías.

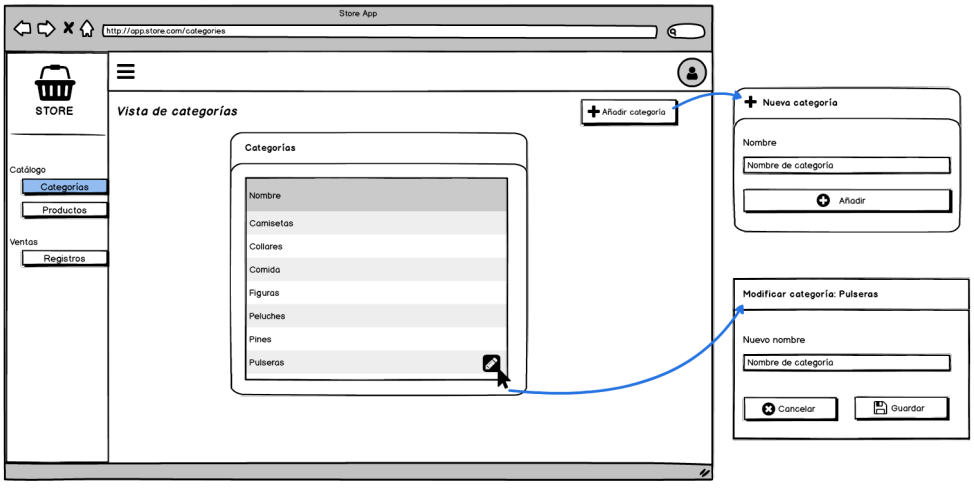


Figura 3.1: Mockup - Vista de categorías

SHSH-8 Añadir producto al catálogo

Los administradores podrán dar de alta nuevos productos en el catálogo. Al igual que ocurre con las categorías estos deben tener un nombre único y cuya comprobación no distinguirá entre mayúsculas y minúsculas. Estos productos tendrán un código de barras asignado, que será alfanumérico y asignado automáticamente durante la creación del mismo.

SHSH-10 Búsqueda sobre productos

Cualquier usuario podrá realizar búsquedas sobre el catálogo, en las que se podrá filtrar por la categoría del producto y una serie de palabras clave. El resultado devuelto contendrá productos pertenecientes a la categoría seleccionada y cuyo nombre contenga alguna de las palabras clave indicadas. Además se podrá indicar también si se desea mostrar exclusivamente los productos activos y el orden de los elementos devueltos, escogiendo ordenar por nombre de producto o su fecha de inserción. Los elementos devueltos muestran un conjunto reducido de atributos, tomando para ello los más relevantes.

En la siguiente figura (3.2) se muestra el mockup diseñado para la búsqueda de productos y el alta de los mismos.

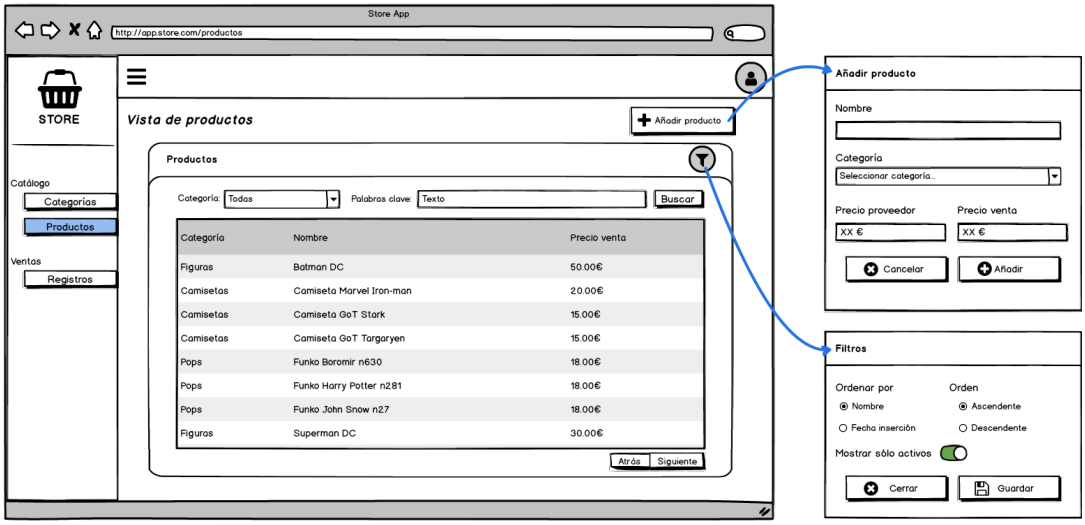


Figura 3.2: Mockup - Vista de productos

SHSH-9 Obtener detalle de producto

Desde los resultados obtenidos en una búsqueda de productos cualquier usuario podrá seleccionar un elemento, el cual llevará a una vista en detalle donde se mostrará toda la información de valor que ofrece dicho producto.

SHSH-11 Modificar datos de producto

Los administradores podrán modificar la información referente a un producto desde su vista en detalle. Se tendrán en cuenta las mismas restricciones detalladas para la inserción relacionadas con el nombre del producto, incluyendo además un control sobre la posible duplicidad en los códigos de barras si estos son modificados.

SHSH-12 Activar y desactivar producto

Se podrá desactivar un producto y volver a activarlo en cualquier momento por parte de los administradores desde la vista en detalle del mismo. Dado que se espera un alto número de productos registrados se podrá priorizar la visualización de los productos activos frente a aquellos de los que no se espera reponer stock.

La figura 3.2 muestra el mockup para la vista en detalle de un producto y las posibles acciones sobre este.

Figura 3.3: Mockup - Vista de detalle de un producto

SHSH-15 Registrar personal

El gerente del sistema podrá agregar usuarios para los empleados que tendrán acceso, cubriendo para ello la información básica de estos, sus credenciales iniciales y los roles en base a sus funciones. Debido a la decisión de que exista un único gerente se controlará que no se introduzca un nuevo usuario con este rol, además todos deben contar con al menos uno de los otros dos roles posibles.

SHSH-16 Listar personal registrado

Desde la vista dedicada al personal, el gerente del sistema podrá observar a todos los usuarios que ha registrado. Para facilitar esta visualización se ofrece un filtro que permita ocultar o mostrar los usuarios bloqueados del listado resultante.

SHSH-17 Actualizar datos del personal

Tanto la información como roles del personal podrán ser actualizados desde el listado de usuarios accesible para el gerente, contando con las mismas restricciones que se vieron durante el registro de estos. Además se impedirá la revocación del rol gerente sobre este usuario especial.

La figura 3.4 muestra el mockup diseñado para la vista de los usuarios del personal, con toda la funcionalidad mencionada para su gestión.

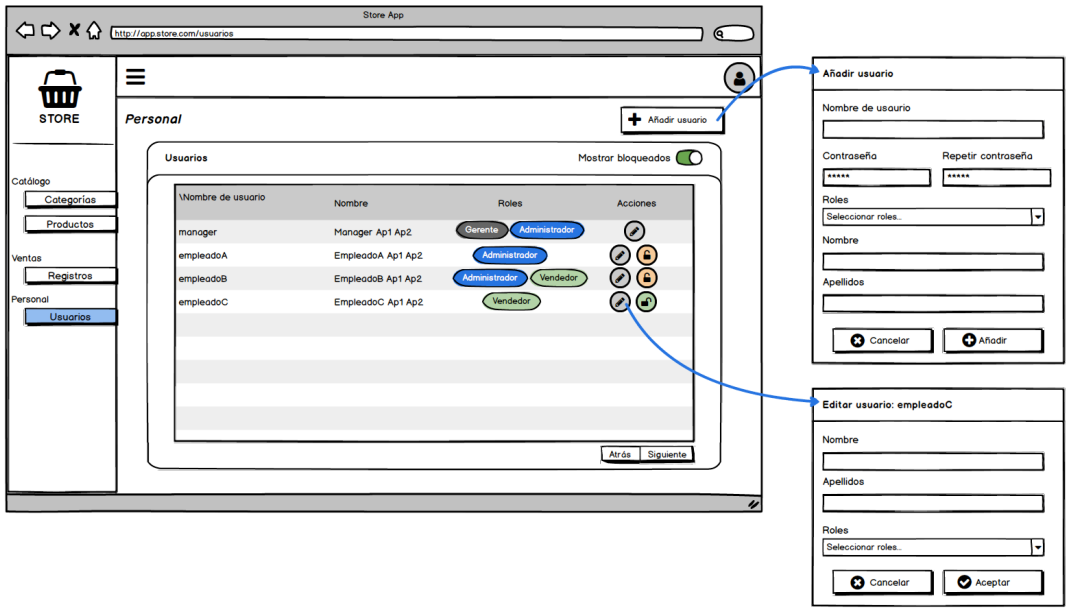


Figura 3.4: Mockup - Vista de personal

SHSH-20 Actualizar catálogo y usuarios en PoS

El terminal punto de venta se actualizará cuando existan cambios en el catálogo o sobre el personal de venta registrado. Para esta función se registrará la fecha de modificación de cada uno de estos elementos, mientras que en el punto de venta se almacenará la marca de tiempo de la última actualización realizada, lo cuál permitirá actualizar solo aquellos elementos que hayan sufrido alguna modificación.

SHSH-21 Identificación en PoS

Una vez cargados los datos en el puntos de venta, los vendedores podrán realizar la identificación en esta aplicación. Esto se permite a través de un proceso de autenticación contra los datos locales que fueron actualizados.

SHSH-18 Registrar venta en PoS

El personal de venta podrá registrar las ventas que realicen durante los eventos en la aplicación PoS. Para ello se gestiona un carrito de compra con los productos de la venta actual junto sus cantidades y costes, incluyendo el importe total a pagar. La primera incorporación de un producto se realizará a través de un buscador con los productos almacenados localmente, y tras agregarlo al carrito se podrá incrementar o quitar unidades de la cesta. Opcionalmente se podrá realizar un descuento sobre las ventas, ya que la tienda puede ofrecer una serie de cupones descuento para promocionarse y atraer clientes potenciales. Al completar la venta se mostrará un calculador del cambio a devolver, en caso de que se pague en efectivo, tras lo cual se generará un identificador único con el que se procede al registro de la venta en el almacenamiento local. Este proceso puede verse reflejado en el mockup de la figura 3.5.

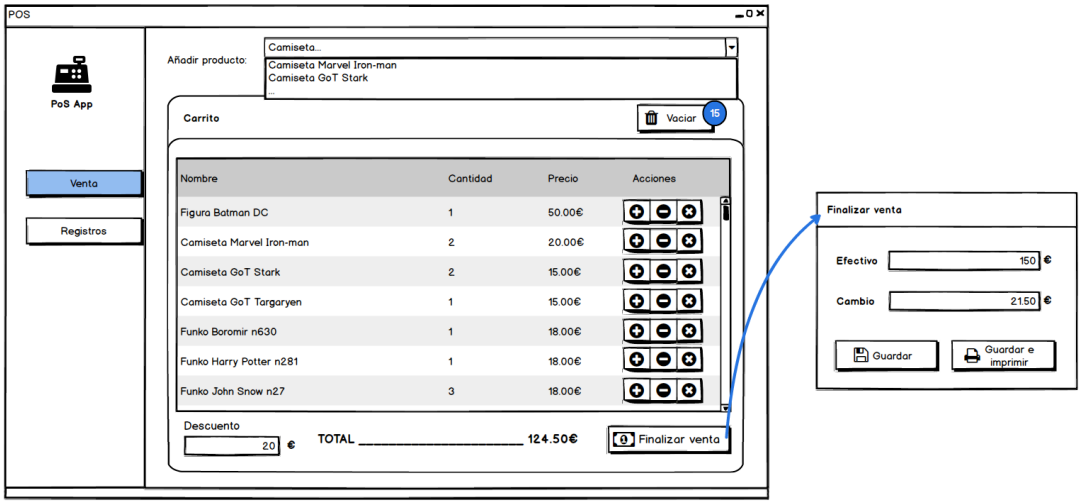


Figura 3.5: Mockup - Vista de venta actual en PoS

SHSH-19 Impresión de ticket de venta

Los vendedores podrán realizar la impresión del ticket de una venta para aquellos clientes que lo soliciten, ya sea durante la finalización de una venta o posteriormente desde la vista de ventas locales registradas. Se requiere la incorporación de elementos visuales que muestren el

estado del enlace con la impresora, con lo que se podrá determinar si la impresión es posible en un determinado momento.

SHSH-22 Búsqueda de ventas registradas en PoS

Desde la vista de registros los vendedores podrán observar las ventas realizadas en un determinado día, calculando además la facturación total de ese día seleccionado. El día actual será seleccionado por defecto la primera vez que se acceda a esta vista durante la sesión. Su mockup correspondiente se refleja en la figura 3.6.

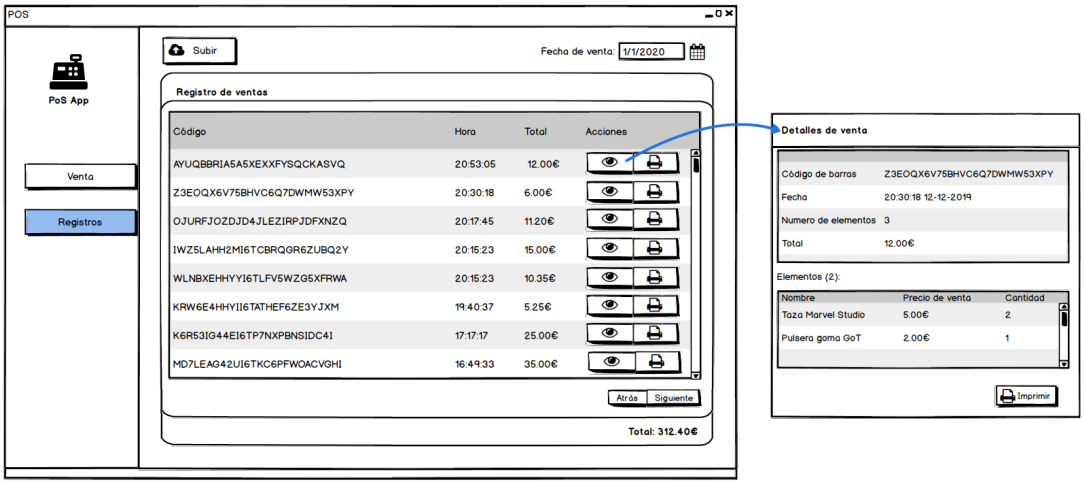


Figura 3.6: Mockup - Vista de ventas registradas en PoS

SHSH-23 Registrar ventas de PoS en servicio

El personal de venta sincronizará las ventas locales en el almacenamiento central del sistema a través del servicio, siendo necesario autenticarse previamente en este. La ventas que se registren satisfactoriamente se marcarán como sincronizadas, para que en futuras sincronizaciones puedan omitirse. Dado que las ventas tienen un identificador universal generado durante su registro local se evita la duplicidad en caso de que sucedan errores durante la sincronización y se reintente este proceso.

SHSH-13 Búsqueda sobre ventas realizadas

Gerente y administradores podrán realizar búsquedas sobre las ventas sincronizadas. Estas búsquedas se realizan en un rango de fechas que por defecto está definido para los últimos 30 días. Además se incluyen los siguientes tres factores de ordenación: fecha, código de venta y cantidad total. Se toma el orden por fecha y descendente por defecto ya que se considera

que será el de uso más frecuente. La siguiente figura (3.7) muestra el mockup para la vista de ventas registradas en el servicio.

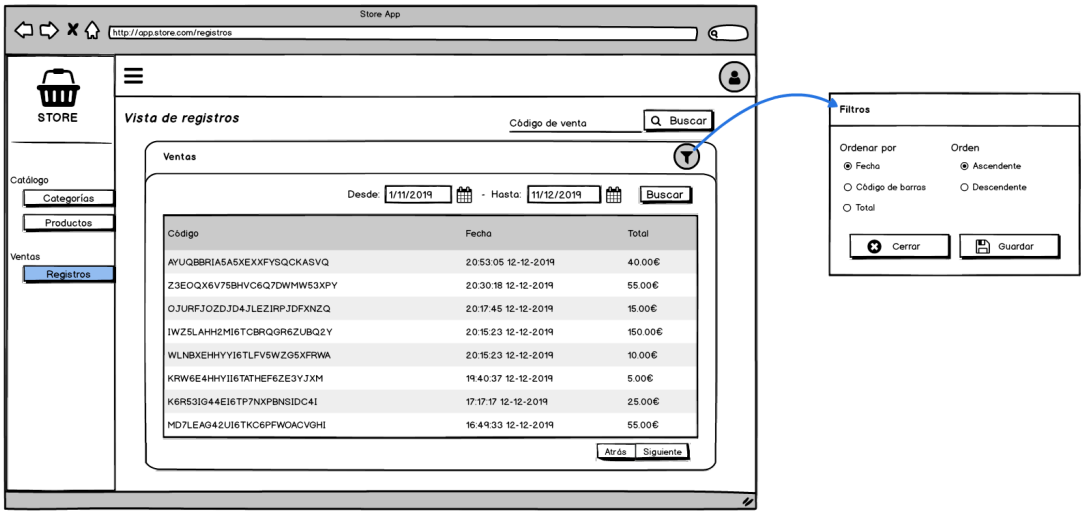


Figura 3.7: Mockup - Vista de ventas

SHSH-27 Obtener detalles de una venta

Desde la vista de ventas, tanto gerente como administradores podrán seleccionar un elemento para inspeccionar en detalle. La vista en detalle de una venta mostrará toda la información relacionada con esta e incluirá un listado con sus lineas de venta, en las que se ofrece un enlace de navegación a la vista en detalle de su respectivo producto. Se muestra el mockup diseñado para esta vista en la figura 3.8.

SHSH-14 Obtener resumenes sobre ventas

Se ofrecerá una página inicial para gerentes y administradores, en las que se mostrarán un resumen referente a las ventas registradas, incluyendo la siguiente información:

- **Últimos 30 días**: número de ventas, número de productos vendidos, total facturado y total de beneficios.
- **Últimos 15 días**: top de productos más vendidos y productos más rentables.
- **Resumen anual** agrupado por meses con su facturación y beneficio correspondiente.

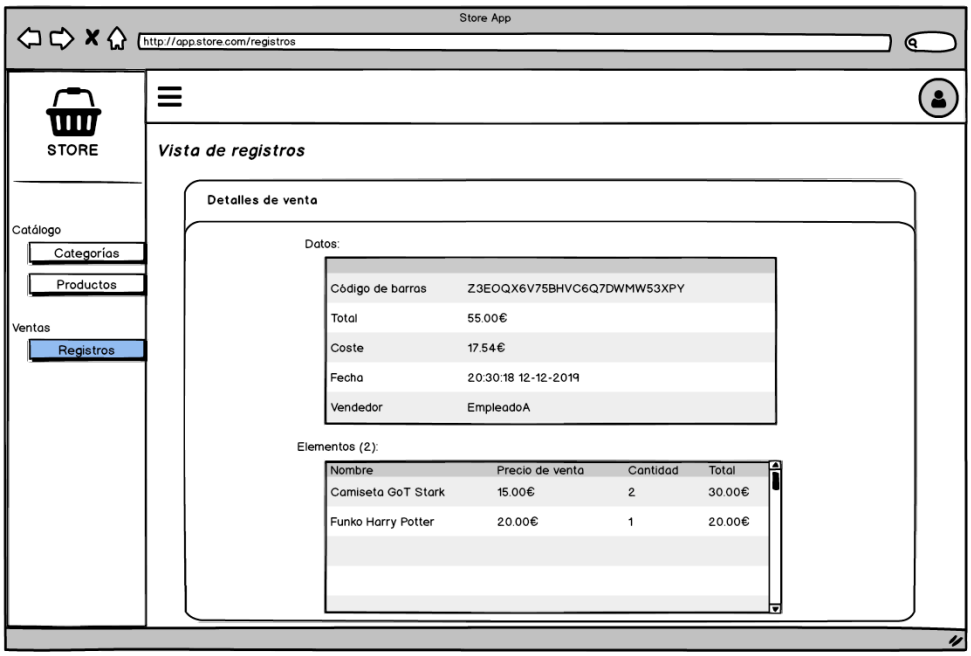


Figura 3.8: Mockup - Vista de detalle de venta

SHSH-24 Autenticación en el sistema

El usuario introducirá sus credenciales para poder acceder al sistema. Una vez autenticado podrá ejecutar toda aquella funcionalidad a la que sus roles den acceso. Durante su actividad un usuario autenticado también podrá elegir cerrar la sesión actual.

SHSH-30 Cambiar credenciales personales

Los usuarios registrados podrán solicitar el cambio de los credenciales con los que se autentican en el sistema. En la aplicación SPA se ofrecerá una vista para la modificación de estos datos, solicitando tanto la nueva contraseña como la actual para confirmar el cambio.

Capítulo 4

Planificación

En este capítulo se aborda la planificación de cada Sprint, siguiendo las pautas marcadas por la metodología detallada en el capítulo 5. Puesto que un factor importante en la priorización de las historias de usuario es la dependencia entre ellas, se realizó un grafo de dependecias para su representación (figura 4.1) que servirá de apoyo durante este proceso.

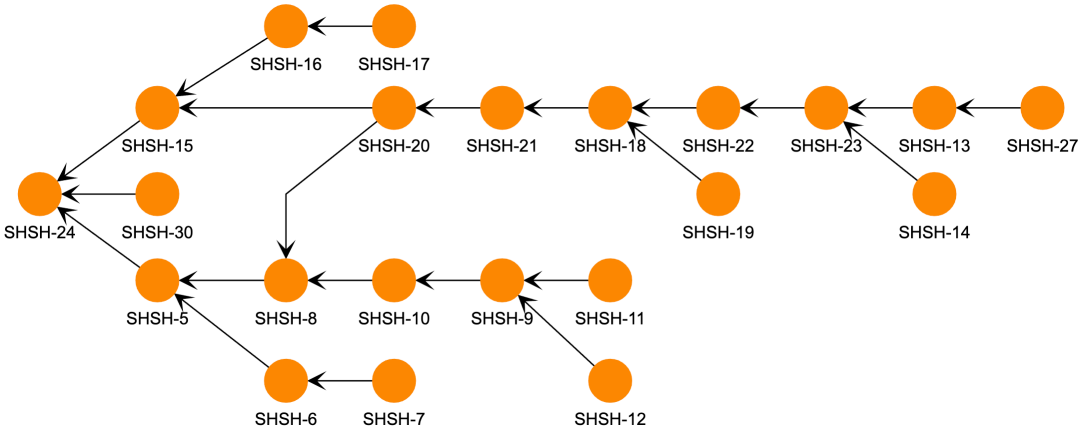


Figura 4.1: Dependencias entre historias de usuario

4.1 Iteraciones

En las iteraciones de desarrollo se podrá observar un valor numérico que indicará los puntos de historia otorgados a ese elemento. Nuestra punto de referencia ha sido la historia de usuario **SHSH-5 Agregar categoría**, contra la que serían comparados el resto de elementos a la hora de otorgar estos puntos.

Sprint 0

Esta iteración tiene un gran componente formativo y de análisis, ya que se centró gran parte del esfuerzo en **estudiar tecnologías y herramientas** con las que nunca se ha trabajo. El resultado final de este sprint inicial no solo es la formación en estas tecnologías, también se preparará el entorno necesario para el desarrollo del proyecto.

Sprint 1

Objetivo: **Gestión de los elementos del catálogo de una tienda**.

Figura 4.2: Pila del sprint 1

Sprint 2

Objetivo: **Gestión de personal y registro de las ventas realizadas en eventos**.

Figura 4.3: Pila del sprint 2

Sprint 3

Objetivo: **Sincronización y visualización de las ventas realizadas, ofreciendo resúmenes visuales que ayudarán al análisis comercial**.

Figura 4.4: Pila del sprint 3

4.2 Duración

En la tabla 4.1 se recoge un resumen del esfuerzo dedicado en cada iteración completada y las fechas en las que se desenvuelve cada una.

Iteración	**Periodos**	**Esfuerzo**
Concepción de la idea del proyecto	15/10/19 - 15/12/20	70h
Sprint 0 - Formación	13/01/20 - 14/02/20	80h
Sprint 1 - Gestión de catálogo	17/02/20 - 13/03/20	68h 16m
Sprint 2 - Gestión de usuarios y registro de ventas	16/03/20 - 03/04/20 13/04/20 - 17/04/20	69h 56m
Sprint 3 - Sincronización y visualización de ventas	27/04/20 - 22/05/20	76h 13m
Elaboración de la memoria	01/06/20 - 04/06/20	80h
Total		**450h 25m**

Tabla 4.1: Resumen de iteraciones

4.3 Recursos materiales

Durante el desarrollo se precisó de un equipo idéntico al que se utilizará en la empresa como servidor del sistema, con un coste de 360€ y con las siguientes características:

- Modelo: Beelink X45 Mini PC
- RAM: LPDDR4 8GB
- Disco: SSD 512GB
- Procesador: Gemini Lake J4105 Quad Core

Además fue necesaria la compra de la impresora que realizará la impresión de los tickets, una impresora térmica EPSON TM-20II por un valor de 128€.

4.4 Análisis del esfuerzo

Uno de los puntos claves de la planificación es la estimación de las tareas. Puesto que a estas se le otorgaba un valor en puntos de historia y que se registró el esfuerzo final dedicado a cada una, se puede realizar una comparativa entre estos dos factores con el que analizar el grado de acierto en las estimaciones realizadas.

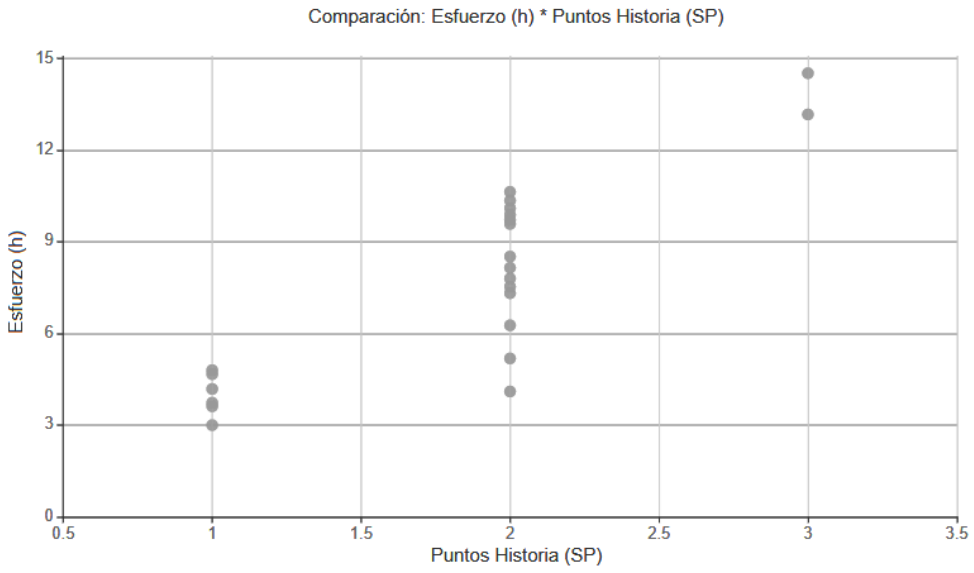


Figura 4.5: Gráfico de dispersión - Esfuerzo Punto de historia

En la gráfica de dispersión de la figura 4.5 se puede observar que, salvo elementos excepcionales, la mayoría de historias de usuario caen en un rango bien diferenciado en base a los

puntos otorgados. Sin contar con los valores atípicos, ya que demuestran un grado de error en la estimación de su complejidad, obtenemos las siguientes medias que demuestran un buen escalado en la valoración de los puntos de historia:

Puntos de historia	**Media**
1 SP	4 h
2 SP	8,14 h
3 SP	13,83 h

Capítulo 5
Metodología

La elección de una metodología que guíe el desarrollo del proyecto conlleva estudiar cuál es la que se podría amoldar mejor a este en base a sus características y realizar un análisis de los beneficios que obtendremos de unas frente a otras. En este caso se ha decidido emplear un enfoque ágil, frente a una metodología más predictiva, por las siguientes razones:

- El **cliente podrá involucrarse** durante todo el ciclo de desarrollo.
- Se entregarán **versiones incrementales** del producto cada cierto tiempo, por lo que el usuario destino podrá adaptarse al sistema poco a poco. Además nos notificarán los posibles errores que se encuentren en las nuevas funcionalidades de cada entrega.
- Obtendremos **feedback** desde las versiones iniciales del producto.
- Facilidad de **adaptación** en caso de introducir cambios en los requisitos entre las distintas iteraciones.
- Durante el desarrollo se emplearán tecnologías con las que no se está familiarizado, por lo que una estimación y planificación detallada sería poco real.

Un referente dentro del enfoque ágil es el marco de trabajo *Scrum* [7], del que tomaremos gran parte de sus elementos existentes. Scrum está ligado al trabajo en equipo, por lo que tendremos que adaptar aquellas partes del proceso que lo necesiten a un trabajo más individual, resultando en una metodología ágil adaptada[1].

5.1 Equipo

Una de las características que definen a estos equipos es que son auto-organizados y multifuncionales, llevarán a cabo su tarea de la manera que mejor convengan y sus miembros

[1] Ron Jeffries, uno de los autores del Agile Manifesto: "¿Te funciona? Entonces utilízalo, pero no lo llames Scrum..."

tendrán las competencias y habilidades necesarias para llevar a cabo el trabajo sin depender de personas externas a este grupo. Los roles empleados son los siguientes:

- **Product Owner**: El Propietario del Producto es uno de los roles clave, ya que es el responsable de maximizar el valor del producto del equipo de desarrollo. Uno de los puntos clave de este rol es el conocimiento sobre el dominio de la aplicación, lo que daría lugar a unos requisitos más completos y con lo que se minimizarán cambios o errores en estos.
- **Development Team**: El equipo de Desarrollo son los miembros destinados a la consecución de los objetivos que marca el Propietario del Producto y los únicos que participan en la creación de los incrementos.

Se omitirá el uso del rol **Scrum Master**, ya que es un rol destinado más a la mediación en el equipo y no encaja en esta adaptación de la metodología, dado que el equipo se compone de un único miembro.

Dada la característica de este equipo, el alumno ejercerá ambos roles durante los diferentes eventos que se verán en la siguiente sección, intentando ser lo más fieles posibles a la posición de cada rol durante su ejecución.

5.2 Eventos

Los eventos son periodos de tiempo limitado (*time-box*), y cada uno está destinado a una finalidad concreta.

5.2.1 Sprint

El *Sprint* es el núcleo de esta metodología que hemos adaptado. Estos se producen en un periodo de tiempo, su duración es constante a lo largo del desarrollo y no suele durar más de un mes. En este periodo no se podrán realizar cambios que afecten al objetivo que se ha definido para dicho Sprint, y al finalizar el mismo se obtiene un incremento del producto utilizable que podrá ser desplegado.

Este evento contiene otros eventos de la metodología que han sido incorporados, como la Planificación del Sprint (*Sprint Planning*), la Revisión del Sprint (*Sprint Review*), la Retrospectiva del Sprint (*Sprint Retrospective*) y el propio trabajo de desarrollo.

5.2.2 Sprint Planning

Evento que se produce al inicio de cada Sprint. El Equipo de Desarrollo, en base al estado de la Pila del Producto, tomará aquellos elementos prioritarios que considere realizables y que se comprometerá a completar en dicho Sprint.

Obtendremos como resultado la definición del Objetivo del Sprint (*Sprint Goal*) y la Pila del Sprint (*Sprint Backlog*), que recoge los elementos de la Pila del Producto escogidos para su realización. Tras esto se procederá a definir en detalle como se construirá la funcionalidad que dará lugar al incremento.

5.2.3 Sprint Review

Una de las etapas finales de cada Sprint consiste en la presentación del incremento creado, donde mostraremos a la parte interesada una demostración de las funcionalidades implementadas.

Finalmente, como Propietario del Producto y en base a los resultados obtenidos, se realizará una revisión de la Pila del Producto, obteniendo información valiosa de la que se sacará provecho en la fase de planificación del siguiente Sprint.

5.2.4 Sprint Retrospective

Este evento pretende revisar la resolución del último Sprint, además de identificar aquellos elementos más significativos que han salido bien y las posibles mejoras a implementar en el desempeño del equipo.

5.3 Artefactos

Los artefactos fueron diseñados para maximizar la transparencia y el registro de la información fundamental del proceso.

5.3.1 Product Backlog

La Pila del Producto es una lista ordenada que contiene las necesidades del producto y expone los requisitos, a un alto nivel, del sistema. El orden marcará la prioridad que se le da a cada elemento, basándonos en el valor de negocio que nos pueda ofrecer actualmente.

Una representación muy extendida para los elementos de esta pila son las Historias de Usuario, donde se describe una funcionalidad que debe incorporarse desde la perspectiva de un usuario del sistema, empleado un lenguaje coloquial y siguiendo una estructura similar a la siguiente:

COMO [*rol del usuario*] QUIERO [*funcionalidad*] PARA [*motivación/beneficio*]

5.3.2 Sprint Backlog

La Pila del Sprint es un subconjunto de los elementos de la Pila del Producto y que han sido seleccionados para su implementación en la iteración actual. El Equipo de Desarrollo es el

encargado de estimar estos elementos y hacer un desglose de los mismos, obteniendo así una lista de tareas a realizar. Las tareas son auto-asignadas por los propios miembros del equipo durante la ejecución del Sprint.

5.3.3 Incremento

Los incrementos están compuestos por los elementos de la Pila del Producto que han sido **completados** durante un Sprint y de los incrementos de Sprints previos. Estos incrementos han de ser funcionales y potencialmente desplegables.

5.4 Aplicación

5.4.1 Proceso

En la figura 5.1 podemos observar el proceso que se ha adaptado de Scrum.

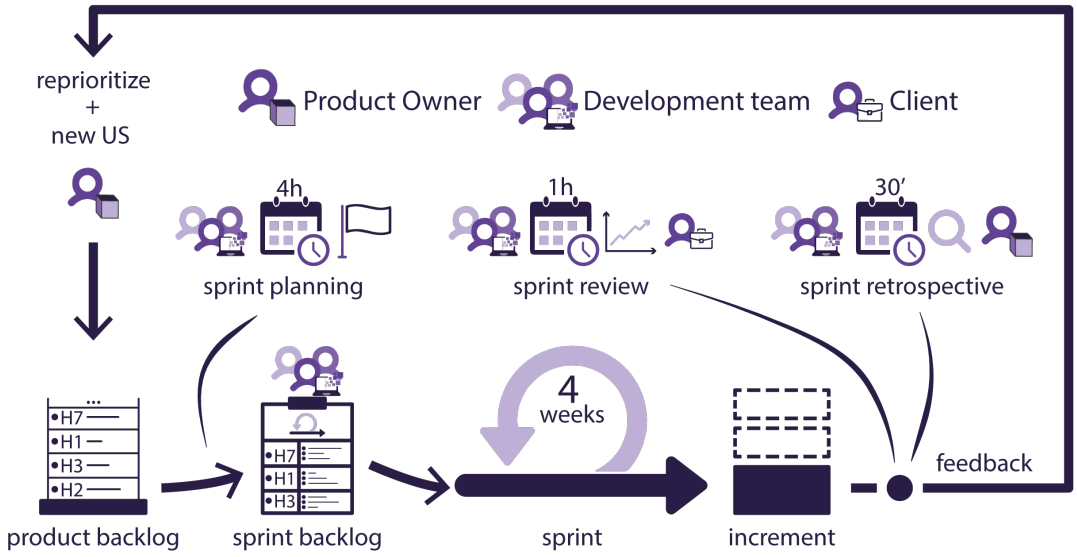


Figura 5.1: Proceso adaptado de Scrum

Estimación

Para la estimación de las Historias de Usuario se han usado los Puntos de Historia (*Story Points - SP*). Los puntos otorgados indicarán la complejidad relativa de un elemento de la pila frente a otro de referencia, por lo que uno de los pasos iniciales será escoger esta referencia base. Los beneficios de esta estimación es que es mucho más sencillo de estimar comparando en tamaños los elementos de nuestra pila, por lo que se reduce el tiempo dedicado a esta tarea.

En la estimación mencionada se ha usado la técnica de Estimación de Póquer (*Planning Poker*), usando la siguiente secuencia de puntos: 0, ½, 1, 2, 3, 5, 8. El motivo tras utilizar una aproximación de la serie de Fibonacci se debe al aumento del error en estimaciones en elementos con puntos más elevados. Tras la estimación, aquellos elementos que tengan un valor alto dentro de la percepción del equipo deberán ser fragmentados en elementos más pequeños, los cuales tendrán valores estimados inferiores y con lo que se minimizará este margen de error.

Diseño del Sprint

Durante la planificación nos apoyaremos en el uso herramientas de diseño vistas en metodologías como PUDS[2]. En esta fase diseñaremos:

- **Prototipos**: Ayudarán a la comprensión de la historia de usuario y podrían relevar detalles interesantes a implementar no considerados en su definición.
- **Diagramas de Casos de Uso**: Gracias a estos no solo podremos definir el flujo de acciones y como debe comportarse el sistema frente a los diferentes actores presentes en él, sino que podremos realizar una estimaciones más precisas en base al tamaño/complejidad de estos.
- **Diagramas de Clases**: Expondrán la estructura del sistema en las diferentes clases necesarias para otorgar la funcionalidad expuesta en los casos de uso.

Flujo de desarrollo

Con el fin de mejorar el proceso de desarrollo y obtener un producto de más calidad se han empleado distintas herramientas que se consideró beneficiosas durante el desarrollo de los incrementos.

La herramienta **Jira** proporcionó las utilidades necesarias para mantener y realizar un seguimiento de los artefactos mencionados en la sección anterior. Además se hizo uso de un tablero *Kanban* durante cada Sprint, donde se podría mover las tareas a través de diferentes estados (**Por hacer**, **En curso**, **Listo**).

Para el desarrollo de estas tareas se hizo uso del flujo de trabajo *GitFlow*, muy aplicable en metodologías ágiles. En este flujo se utilizan una serie de ramas en el repositorio del proyecto, cada una con un propósito específico, y con las que se pretende dar una mejor organización en el flujo de la integración continua. En la figura 5.2 vemos una representación del uso de las diferentes ramas, donde cada *feature* se corresponderá con una de las historias de usuario.

[2]PUDS: Proceso Unificado de Desarrollo del Software

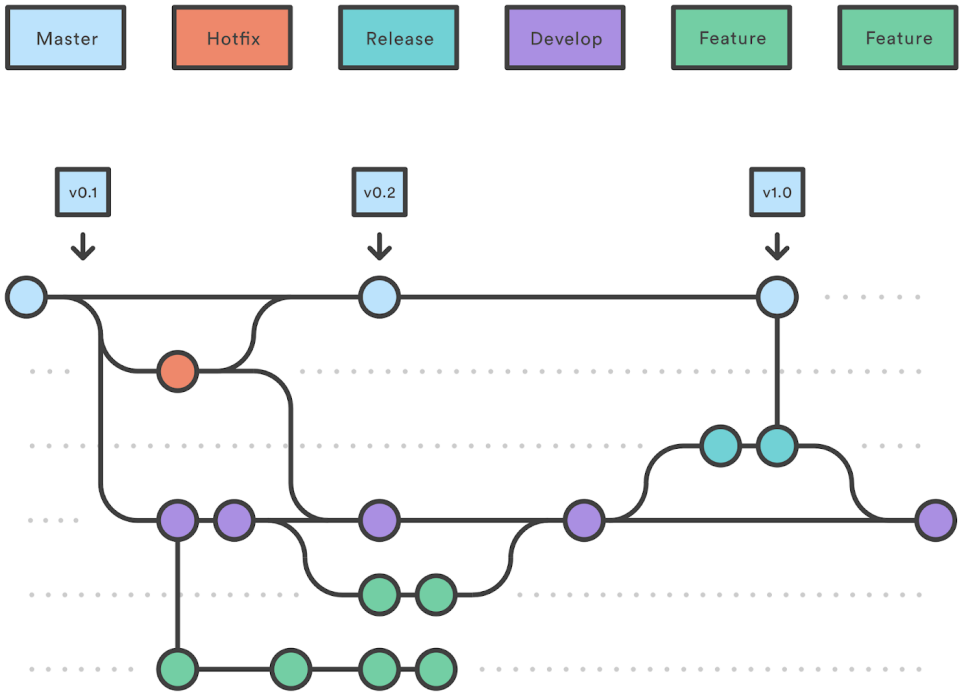


Figura 5.2: Ejemplo de uso de ramas con GitFlow

El uso de este flujo facilitaría después la aplicación de herramientas de integración continua (**Jenkins**) e inspección continua (**Sonar**), que actuarán sobre cada una de estas ramas. Para dar por finalizada la implementación de una de las tareas se debe superar las pruebas en el entorno de integración y cumplir con los criterios de calidad mínimos marcados por la herramienta de inspección.

Capítulo 6

Fundamentos tecnológicos

En este capítulo detallaremos las tecnologías y herramientas utilizadas en los tres componentes principales de nuestro sistema y aquellas que dan soporte al proceso de desarrollo.

6.1 Tecnologías empleadas en el servicio

Spring

Spring [8] es un framework de código abierto para el desarrollo de aplicaciones empresariales en la plataforma Java. Su ecosistema de proyectos brinda todo tipo de utilidades necesarias para un desarrollo ágil de este tipo de aplicaciones, entre las que destacaremos:

- **Spring Boot** [9]: Proyecto destinado a simplificar la configuración de la aplicación y poder centrarse más en el desarrollo. Además simplifica el uso de dependencias Maven a través de paquetes denominados *starters*, que incluye todas las dependencias necesarias para ciertos aspectos de la aplicación (web, seguridad, acceso a datos, ...). Otro de los puntos extra es que eliminará posibles conflictos entre las dependencias transitivas de los distintos paquetes starters que sean utilizados.

- **Spring Security** [10]: Permitirá gestionar todo lo relativo a la seguridad de la aplicación web. Una de las características que proporciona es el uso de roles de usuario, con los que podremos definir una serie de reglas de acceso a los recursos de la aplicación.

- **Spring Data JPA** [11]: Su objetivo es unificar y facilitar el acceso a distintos tipos de tecnologías de persistencia, tanto a bases de datos relacionales como *NoSQL*. Simplifica la la implementación de la capa de acceso a datos proporcionando soporte para JDBC, Hibernate, JPA, etc.

Rest

REST es una arquitectura de desarrollo web que puede ser utilizada en cualquier cliente que utilice HTTP para obtener *recursos* (elementos de información) o indicar la ejecución de operaciones sobre estos recursos, empleando un formato determinado para el intercambio de datos como pueden ser XML o JSON.

Maven

Apache Maven [12] es una herramienta para la gestión, construcción y comprensión de proyectos Java. Se basa en el concepto de *Project Object Model (POM)*, un fichero XML del que sacará toda la información del proyecto a construir.

En este fichero POM se definirán las dependencias necesarias del proyecto, los distintos perfiles que dan soporte a configuraciones alternativas y los distintos plugins que se acoplan a las fases del ciclo de vida (como puede ser la generación de un informe de cobertura de código en la ejecución de pruebas).

MySQL

MySQL [13] es un sistema gestor de bases de datos relacional. MySQL es una opción razonable para ser usado en el ámbito empresarial, al estar basado en código abierto permite a pequeñas empresas y desarrolladores disponer de una solución fiable y estandarizada para sus aplicaciones.

Flyway

Flyway [14] es una herramienta de código abierto para la migración de base de datos. Soporta un gran número de bases de datos como Oracle, SQL Server, DB2, MySQL, MariaDB, PostgreSQL, Redshift, H2. Además ofrece una serie de plugins para su integración con Spring Boot, Grails y otros frameworks.

Dado que el modelo de datos relacional irá evolucionando a lo largo del desarrollo, esta herramienta ofrece la funcionalidad necesaria para automatizar la aplicación de estos cambios. Flyway creará la tabla *SCHEMA_VERSION* y en esta almacenará el histórico de las migraciones realizadas en base a los *scripts* que le proporcionemos, junto con el resultado de cada ejecución. La figura 6.1 muestra una representación de su funcionamiento.

JUnit

JUnit [15] es un framework de código abierto para la automatización de las pruebas (tanto unitarias como de integración) en proyectos Java. Provee de las herramientas, clases y mé-

todos que facilitan la tarea de realizar pruebas en el sistema y asegurar su consistencia y funcionalidad.

6.2 Tecnologías empleadas en la web SPA

React

React [16] es una librería JavaScript de código abierto que permite el desarrollo de interfaces de usuario de forma sencilla. Sigue el enfoque basado en componentes que ayuda a construir componentes del interfaz de usuario reutilizables.

React aporta una serie de claras ventajas frente a la forma clásica de realizar una web, ya que sus facilidades para el desarrollo unido al rendimiento, flexibilidad y clara organización del código la convierten en una gran opción.

Redux

Redux [17] es un contenedor predecible del estado de aplicaciones JavaScript. Redux se encarga en cierta manera de desacoplar el estado global de la aplicación web frontal de la parte visual, es decir, los componentes. Con su incorporación nos beneficiaremos de una arquitectura escalable de datos y con un mejor control sobre el flujo de datos y el estado de la aplicación. Estas ventajas son favorables para aplicaciones que tienen cambios constantes (aplicaciones medianas y grandes), puesto que Redux ayuda a mantener su complejidad bajo control.

Se puede usar Redux combinado con React, o cualquier otra librería de vistas. Es muy pequeño y no tiene dependencias.

Material-UI

Material-UI [18] es una librería que nos proporciona una serie de componentes React para agilizar el desarrollo. La implementación de estos componentes sigue la normativa de diseño que expone Material Design [19], donde tocan temas como la disposición de los elementos, elección de colores (proporcionando incluso herramientas para ello [20]), aspectos tipográficos y diseño de temas personalizados.

El uso de esta librería nos proporcionara un aspecto estandarizado y con un acabado muy logrado en nuestras aplicaciones.

Jest

Jest [21] es un framework de pruebas para JavaScript. Dentro de las pruebas que ofrece encontraremos los test de instantáneas o *Snapshot Test* [22]. Su funcionamiento consiste en

capturar el resultado del renderizado de los componentes React y realizar una comprobación con capturas previas para ver si han sufrido algún cambio.

Dado que gran parte de los componentes diseñados emplearán el estado Redux, y que estos suelen variar dependiendo del estado en el que se encuentren, podremos realizar test de captura para comprobar que los resultado del renderizado en los diferentes estados que afecten al componente es el deseado.

npm

Node Package Manager [23], o simplemente npm, es el gestor de paquetes JavaScript de NodeJS por excelencia. Gracias a él podremos disponer de cualquier librería de su repositorio y nos permitirá agregar dependencias de forma simple, distribuir paquetes y administrar eficazmente tanto los módulos como el proyecto a desarrollar en general. Para ello utiliza el archivo `package.json`, donde almacena todos estos datos relevantes de la aplicación.

6.3 Tecnologías empleadas en la aplicación PoS

Electron

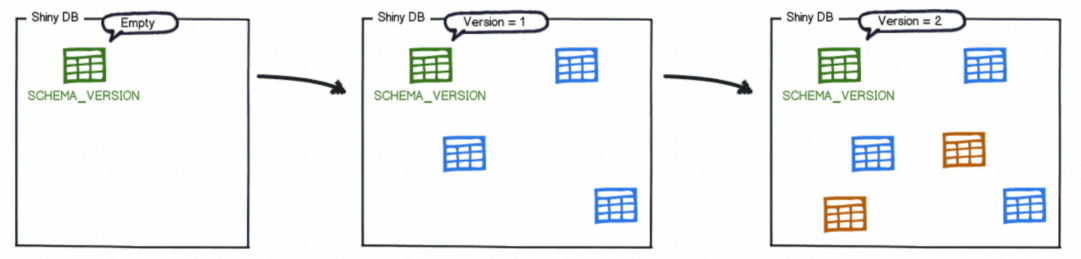
ElectronJS [24] es un framework que permite desarrollar aplicaciones de escritorio multiplataforma utilizando tecnologías web. En Electron existe un proceso único que ejecuta el *script main* del archivo `package.json`, llamado proceso principal. El *script* que corre en el proceso principal puede mostrar una interfaz gráfica creando páginas web. Podemos ver una representación de su arquitectura en la figura 6.2.

Debido a que Electron emplea Chromium para mostrar páginas web se da uso de la arquitectura multi-proceso de Chromium. Cada página web en Electron corre en su propio proceso, el cual es llamado el proceso renderizador o *renderer*. Además, hay varias formas de comunicar el proceso principal y el proceso renderizador, como los módulos `ipcRenderer` e `ipcMain` para enviar mensajes entre estos y el módulo `remote` para una comunicación de tipo *RPC*.

Dado que el desarrollo del proceso renderizador de nuestra aplicación consistirá en una página web SPA, se utilizarán las tecnologías vistas en la sección 6.2.

PouchDB

PouchDB [25] es una base de datos *NoSQL* de navegador que permite a las aplicaciones web almacenar datos de manera local. Estos datos serán almacenados en forma de documento y tendrán un identificador único (`_id`). PouchDB nos ofrece un API (Application Programming Interface) [26] asíncrona con la que ejecutaremos las diferentes operaciones.



installed_rank	version	description	type	script	checksum	installed_by	installed_on	execution_time	success
1	1	Initial Setup	SQL	V1__Initial_Setup.sql	1996767037	axel	2016-02-04 22:23:00.0	546	true
2	2	First Changes	SQL	V2__First_Changes.sql	1279644856	axel	2016-02-06 09:18:00.0	127	true

Figura 6.1: Ejemplo básico del funcionamiento de Flyway

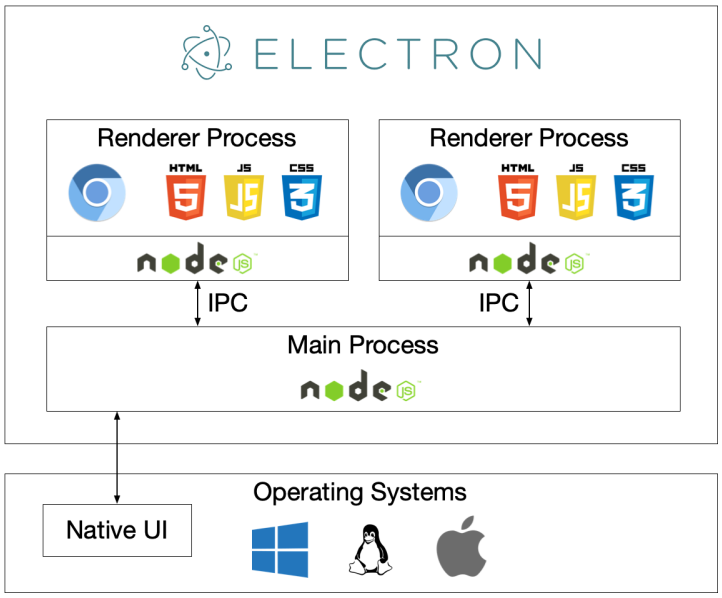


Figura 6.2: Arquitectura de Electron JS

Electron Forge

Electron Forge [27] es una de las herramientas existentes para la construcción de aplicaciones Electron. Gracias a su interface de línea de comandos (CLI - Command line interface) se podrá ejecutar, probar y empaquetar nuestra aplicación Electron. Además, proporciona los recursos necesarios para generar los diferentes instalables específicos de cada plataforma.

6.4 Herramientas de apoyo a la metodología y el desarrollo

Git

Git [28] es un sistema de control de versiones (SCM - Source Code Management System) distribuido. Fue diseñado pensando para manejar cualquier tamaño de proyecto con velocidad y eficiencia.

Las principales ventajas sobre otros herramientas SCM son la utilización de un áreas de preparación, con los cambios aceptados para confirmación, y la adaptación a múltiples flujos de trabajo gracias a sus sistema de ramas.

GitHub

GitHub [29] es una plataforma que permite alojar proyectos que utilicen el sistema de control de versiones Git. Estos podrán ser alojados en repositorios públicos o privados, siendo estos últimos limitados en número para cuentas gratuitas.

Jenkins

Jenkins [30] es un servidor de integración continua (CI). Permite la creación de trabajos que se ejecutarán sobre las nuevas versiones del código. Estos trabajos generalmente automatizan el proceso de compilación y ejecución de pruebas, aunque se puede agregar otros pasos extra como la ejecución de revisores de código, construcción de artefactos, etc.

Una de las ventajas de su incorporación es la pronta detección de errores a lo largo del desarrollo y su reducción en fases de puesta en producción. Al ejecutar nuestra aplicación en entornos ajenos al desarrollo detectamos posibles errores que sucederían durante la implantación de estos, por ello se recomienda usar entornos con las mismas características que los equipos finales.

SonarQube

SonarQube [31] es una plataforma de inspección continua que evaluará nuestro código. Para ello empleará una serie de herramientas de análisis estático de código fuente y con las que se obtendrán unas métricas de la calidad del mismo.

Nos informará sobre código duplicado, estándares de codificación, pruebas unitarias, cobertura de código, complejidad ciclomática, errores potenciales, comentarios y diseño del software.

Jira

Jira [32] es una herramienta para la administración de tareas de un proyecto. En principio Jira se diseñó como un gestor de incidencias y errores, sin embargo se ha convertido en una poderosa herramienta de gestión de trabajo para todo tipo de casos de uso, desde la gestión de requisitos y casos de prueba hasta el desarrollo de software ágil.

Una de las opciones que ofrece es la creación de proyectos en base a unas plantillas. Entre las disponibles hay una dedicada a Scrum en la se podrá registrar los artefactos de esta metodología.

Este producto se ofrece de manera gratuita para equipos de 10 o menos integrantes.

IntellJ IDEA

IntellJ IDEA [33] es un entorno de desarrollo integrado (IDE - Integrated Development Environment) multiplataforma. Permite el uso de una gran cantidad de lenguajes y ofrece utilidades para distintos frameworks, haciendo posible el desarrollo en un único entorno.

Entre sus principales ventajas se destaca:

- Gran navegabilidad a través del código.
- Incorporación de herramientas para bases de datos con la que poder realizar conexiones, ver sus esquemas y realizar consultas.
- Refactorizaciones de lenguaje cruzado.

Balsamiq + Draw.io + StarUML

Balsamiq [34], Draw.io [35] y StarUML[36] has sido las herramientas utilizadas para la creación de prototipos, diagramas de casos de uso y diagramas de clases.

Capítulo 7

Desarrollo

Con este capítulo se pretende detallar los aspectos más relevantes, así como la toma de decisiones, durante el proceso de desarrollo de este sistema.

7.1 Sprint 0

En este Sprint inicial habrá tres objetivos principales:

- Configurar las herramientas necesarias y el entorno de pruebas.
- Creación del arquetipo de los componentes del sistema (servicio, web SPA y PoS).
- Tomar un contacto inicial con las tecnologías con las que no se está familiarizado.

7.1.1 Arquetipos y fase de capacitación

En esta etapa inicial se produce la creación de los tres arquetipos iniciales, que serán la base para el desarrollo de los tres componentes del sistema.

Arquetipo del servicio

El arquetipo del servicio será un proyecto Maven, construido en base a uno similar que compartía el uso de varias tecnologías, como Spring Boot, Spring MVC, Spring Data JPA, Spring Security, JUnit y MySQL. El diseño de este arquetipo seguiría la misma estructura dividida en tres capas del proyecto de referencia (servicios/presentación, lógica de negocio y acceso a datos; representadas en la figura 7.1), donde se definían pruebas de integración para la capa de lógica de negocio y pruebas de unidad para la capa de datos.

Para aumentar el alcance de las pruebas se investigó la incorporación de pruebas automatizadas sobre la capa de servicios REST. Navegando por la documentación de la página de

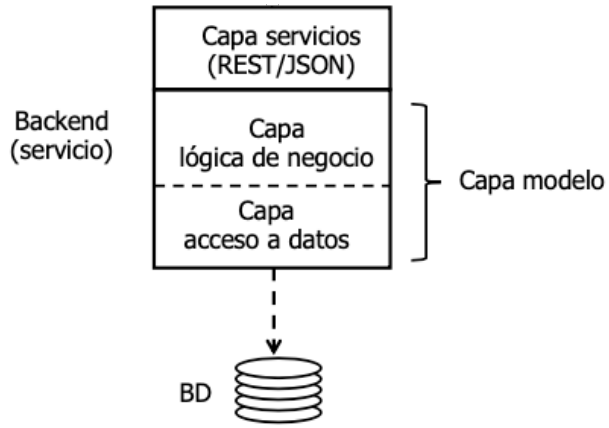


Figura 7.1: Arquitectura del servicio en 3 capas

Spring se encontró una destinada a las pruebas de los componentes web [37], donde se ofrece una guía de incorporación de este tipo de pruebas a la aplicación.

Finalmente se incorporó las dependencias necesarias para agregar la herramienta de migración de bases de datos Flyway, con la que se podrá automatizar los cambios en el modelo de la base de datos durante los despliegues.

Arquetipo de aplicación web SPA

Para la creación de este arquetipo se utilizará la herramienta `create-react-app` [38], un CLI oficial de React con la que se puede crear rápidamente una aplicación sin la necesidad de configuraciones extra. A través de un simple comando se generará un proyecto JavaScript con React, Webpack, Babel y otras utilidades base, configurado y listo para trabajar.

Al proyecto generado se le incorporan las diferentes tecnologías escogidas para el desarrollo de la web SPA, como son la librería Redux y el framework Material-UI.

El siguiente paso realizado fue la creación del tema de la web SPA. Para ello se seguirían las instrucciones expuestas en la página de Material-UI sobre la personalización [39] de temas, definiendo los colores principales que usaría la aplicación, modificaciones de estilos a aplicar sobre los componentes del framework, etc. Finalmente se crearon los componentes globales de nuestra web SPA, entre los que podemos incluir la barra superior, la barra de navegación y el contenedor de las diferentes vistas.

La estructura resultante se refleja en la figura 7.2, que detallaremos a continuación:

- **assets**: Reune los recursos de la aplicación, como imágenes, fuentes de texto, etc.
- **backend**: Contiene la funcionalidad para realizar las llamadas al servicio.

- **i18n**: Archivos de configuración y mensajes en los dos idiomas disponibles de la aplicación (castellano e inglés).
- **modules**: División principal de la aplicación SPA. Cada módulo está pensado para agrupar los componentes por funcionalidades o rutas de la aplicación.
  - **components**: Contenedor de los componentes React.
  - **constants**: Constantes y enumerados ligados al dominio del módulo.
  - **styles**: Estilos definidos para los componentes.
  - **tests**: Pruebas de instantanea de los componentes del módulo.
  - **actions.js, actionTypes.js, reducer.js y selectors.js**: Archivos destinados a controlar la proporción del estado correspondiente de cada módulo.
- **store**: Configuración del estado Redux y funcionalidad para crear el árbol de estado que contendrá las división del estado repartido por los diferentes módulos.
- **theme**: Contiene las modificaciones de los estilos del tema que se aplica a los componentes de Material UI, además de definiciones adicionales.
- **utils**: Destino de la funcionalidad más general de la aplicación, como funciones para la validación de formularios, fórmulas, etc.
- **index.js**: Este fichero ejecutará toda la configuración previa necesaria para el renderizado de nuestra web SPA. En él se define cuál es el componente a renderizar y dónde debe renderizarse.
- **.env.development y .env.production**: En estos archivos se podrá definir variables para el entorno de desarrollo y el entorno de producción respectivamente.

Arquetipo de aplicación PoS

Gracias a la herramienta CLI proporcionada por Electron Forge se podrá crear un proyecto Electron a partir de diferentes plantillas. En este casó se utilizó la plantilla que incorpora el empaquetador Webpack preconfigurado [40] y que contendrá los siguientes ficheros:

- **src\main.js**: Archivo que invoca el proceso principal de Electron y en el que se definirán las ventanas y las respectivas aplicaciones web que estas ejecutarán.
- **src\renderer.js**: Es el equivalente al archivo index.js de nuestra web SPA. Será ejecutado por un proceso renderizador de Electron.

Figura 7.2: Estructura de la aplicación web SPA

- **webpack.main.config.js**: Contiene la configuración Webpack del archivo main.js.
- **webpack.renderer.config.js**: Contiene la configuración Webpack del archivo renderer.js.
- **webpack.rules.js**: Contiene reglas comunes a distintas configuraciones. Este archivo será invocado por las configuraciones que requieran estas reglas, como es el caso de las dos configuraciones previas.
- **package.json**: Contiene las propiedades y dependencias necesarias del proyecto. Adicionalmente en él se podrá agregar los comandos que ejecutaran los *script* de creación de los instalables de la aplicación.

Puesto que la aplicación Electron ejecutará una aplicación web SPA, se incorporó el arquetipo construido previamente a este, añadiéndole una página de autenticación. El resultado puede verse en la figura 7.3.

Finalmente se agregó las dependencias de creación de bases de datos documentales PouchDB, implementando la funcionalidad para la creación de un objeto donde se definían unos métodos CRUD genéricos (figura 7.4) y que se integrará posteriormente con la técnica de *destructuring* [41, pág. 86] en las diferentes bases de datos que se definan.

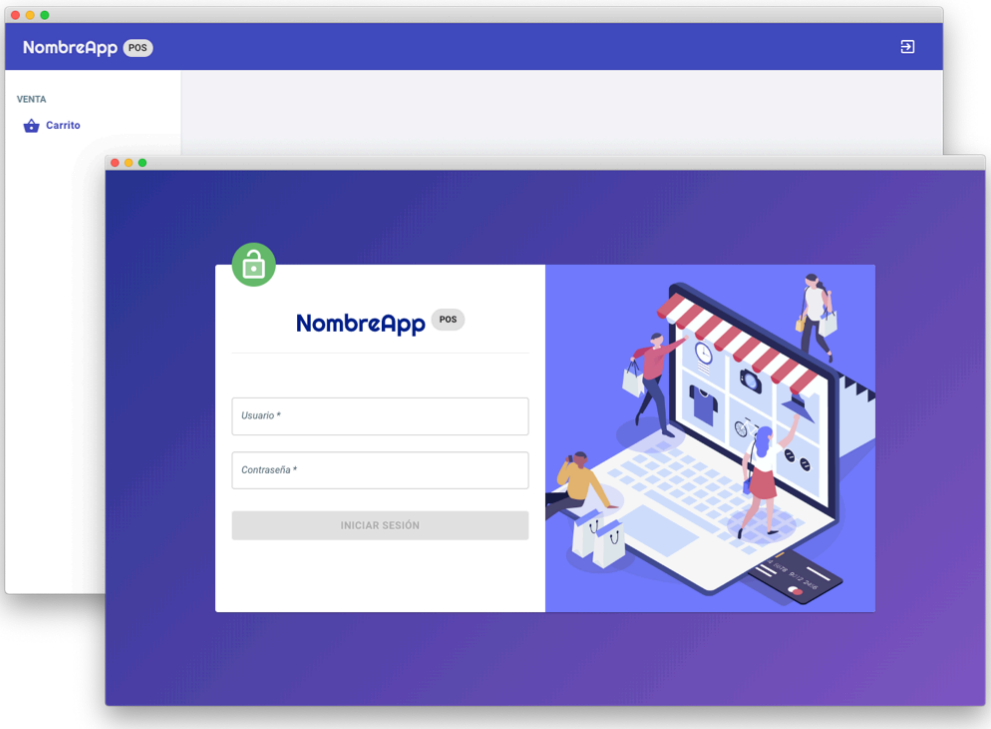


Figura 7.3: Arquetipo de la aplicación PoS

```
// Create
const add = db => async (doc) => {...};
// Read
const getById = db => async id => {...};
const getAll = db => async () => {...};
// Update
const update = db => async (id, data) => {...};
// Delete
const remove = db => async id => {...};

const PouchCRUD = db => ({
    add: add(db),
    getById: getById(db),
    getAll: getAll(db),
    update: update(db),
    remove: remove(db),
});
```

Figura 7.4: Objeto con métodos CRUD genéricos para PouchDB

Gracias a la construcción de este arquetipo se tomó contacto con las tecnologías menos conocidas:

- Creación de una aplicación Electron, observando sus características y funcionamiento.
- Uso de bases de datos PouchDB, cuya funcionalidad proporcionada por su librería para la interacción con estas es totalmente asíncrona.

7.1.2 Creación de proyecto y configuración de entorno

A continuación detallaremos los pasos realizados durante el proceso de creación del proyecto y de configuración de los entornos:

1. Creación del repositorio en la plataforma GitHub donde se alojará el código fuente del proyecto. Este repositorio estará dividido en tres partes: **shiftshop-service**, **shiftshop-spa** y **shiftshop-pos**; y en los que se incorporarán como base de cada uno su respectivo arquetipo.
2. Creación del proyecto en la herramienta Jira, usando para ello la plantilla de proyectos ágiles Scrum y con la que podremos registrar los artefactos de la metodología adaptada.
3. Configuración del entorno de integración e inspección. Para ello se utilizó un equipo en propiedad que reside en el hogar personal. Tras esto se procedió con la instalación de las herramientas Jenkis [42] y SonarQube [43], siguiendo las guías que ofrecen.
4. Conexión de IntelliJ IDEA y Jira [44]. Con esta integración se obtendrán las tareas del proyecto Jira en una vista del IDE y se podrá realizar un seguimiento del tiempo empleado en cada tarea gracias a la herramienta Time Tracking (7.5).

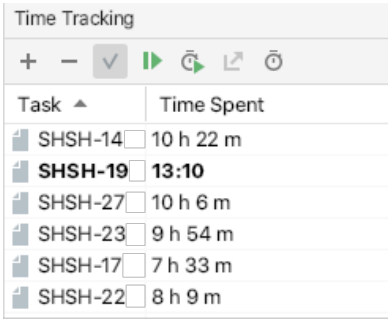


Figura 7.5: Herrramienta de seguimiento de tareas en IDE IntelliJ

5. Conexión de GitHub y Jira [45]. Conectando estas dos herramientas se podrá enlazar las tareas con sus *commits* asociados como puede observarse en la figura 7.6. Para realizar

este enlace se utilizaran los Smart Commits [46], en los que se indicará el identificador de la tarea seguido de unos comandos en los que se podrá indicar un comentario, el registro de tiempo y una transición del estado de la tarea.

Figura 7.6: Tarea de Jira con sus commits asociados del repositorio de GitHub

6. Finalmente se creará un trabajo en Jenkis, que será el encargado del proceso de integración continua del proyecto. Se conectará este trabajo con el repositorio de GitHub que contiene el código fuente gracias a los plugins de los que dispone Jenkins, y que se ejecutará cada vez que detecte un cambio en alguna de las ramas que se le indiquen. Dado que el servidor de integración es un equipo privado, para realizar las comprobaciones de cambios en las ramas se configura el trabajo para que consulte la dirección del repositorio cada 15 minutos. Finalmente se establece la ejecución en orden de las siguientes acciones:

    - Ejecución de la meta **install** de maven sobre el servicio.
    - Inspección del código con SonarQube.
    - Ejecución de las pruebas de la aplicación web SPA, seguido de la compilación optimizada de la aplicación y el empaquetado de esta en un archivo .zip.
    - Ejecución de las pruebas de la aplicación PoS, seguido de la construcción de los instalables para las tres plataformas disponibles (Windows: .exe, MacOs: .zip y Linux: .dbm).
    - Guardar los archivos generados (.jar con el servicio ejecutable, .zip con la web SPA optimizada y los diferentes instalables generados)

7.2 Sprint 1

7.2.1 Análisis

El objetivo marcado para este Sprint fue el de ofrecer la funcionalidad para **registrar y gestionar el catálogo de una tienda**. Con esto en mente se realizaron los primeros prototipos de la aplicación web SPA, los cuales incorporarían todas aquellas historias de usuario que tuviesen relación con el objetivo definido.

Tras el diseño de los prototipos se procedió con un desglose en casos de uso, donde se podrá ver las capacidades de cada rol en el sistema y el flujo de las acciones que estos invocan. En este flujo se puede observar la relación de las historia de usuario con una porción del flujo, englobando uno o más casos de uso para completar dicha funcionalidad. Los prototipos pueden observarse en el capítulo 3.2, mientras que la figura 7.7 representa uno de los diagramas de caso de uso obtenidos.

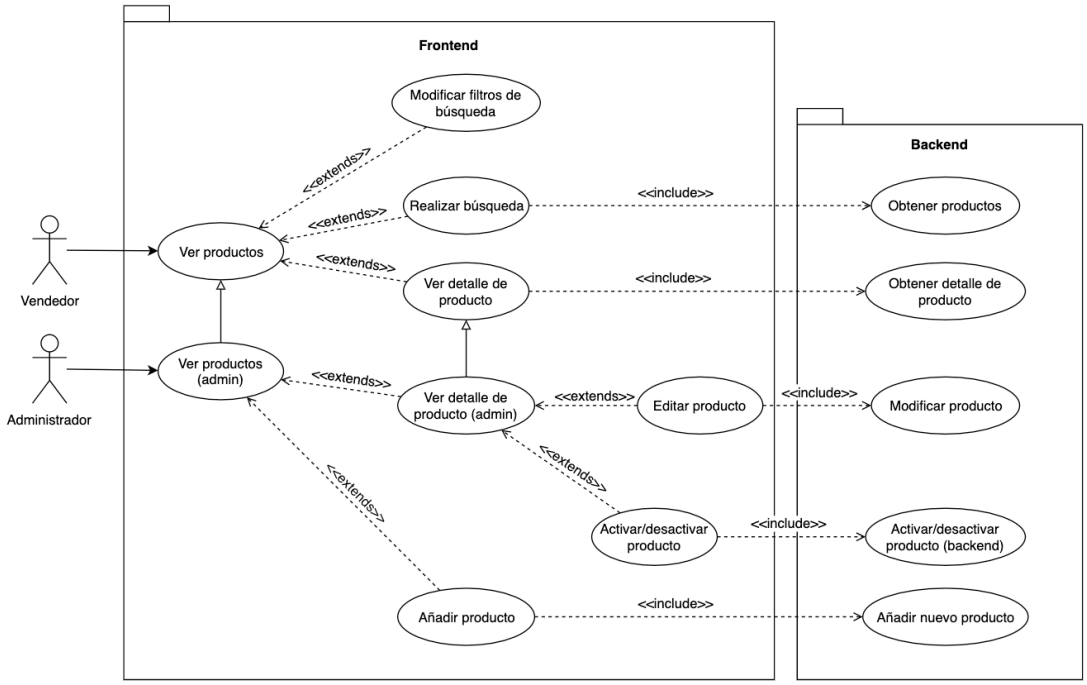


Figura 7.7: Casos de uso para la gestión de productos

Gracias a este proceso de desglose se pudo realizar una estimación más precisa, ya que el margen de error que se puede producir al estimar elementos cada vez más pequeños siempre será inferior. Finalmente se recogieron todas las historias de usuario que se incorporarían a este Sprint, detalladas en el capítulo 4.1.

Durante todo este proceso se incluyeron los detalles de cada historia de usuario en su respectiva representación de Jira, incluyendo **detalles técnicos sobre lo que se haría** y **pruebas a tener en consideración**, obtenidas del ejercicio del análisis.

7.2.2 Diseño e implementación

Mediante el diseño de este Sprint se investigaron ciertos aspectos que estarían presentes no solo en gran parte de las tareas actuales, sino también de futuros sprints, como:

- **Tratamiento de los errores**: Los errores capturados durante la invocación de los métodos con la lógica de negocio serán tratados y devueltos por los controladores, localizándolos para los dos idiomas disponibles de la aplicación. A través de la anotación de Spring `@ExceptionHandler` se indica el método del controlador que capturará dicha excepción, mientras que con la anotación `@ResponseStatus` se indica el código HTTP a devolver.

  En estos mensajes podemos diferenciar dos tipologías, una destinada a la validación de los parámetros de entrada de los controladores, y otra con un aspecto más general destinada al control de errores más relacionados con la propia la lógica de negocio, como puede ser la inserción de productos con nombres duplicados.

- **Conversión de mensajes a JSON**: La conversión de las entradas y salidas en el controlador se realizará gracias a la librería Jackson, a través de la cual se transformará un objeto Java a su respectiva representación JSON y viceversa. Durante esta conversión entran en escena las utilidades de validación en los controladores del servicio, utilizando para ello características del framework Spring Web y del paquete `javax.annotation`.

- **Notificación de resultados**: Se decidió incluir notificaciones temporales cuando se desea informar al usuario del resultado de una petición, como pueden ser la correcta inserción de una categoría, su modificación, etc. Para este aspecto se incorporó la librería **notistack** [47].

- **Imágenes de marcador de posición**: Cuando no existen resultados que mostrar en una búsqueda se suele indicar con un mensaje visual en la interface de usuario, pero incluir una imagen visual suele ser más atractivo. Se empleó para ello las ilustraciones de la colección open-source **unDraw** [48].

Las siguientes figuras muestran un resumen del diseño resultante y las entidades de este sprint:

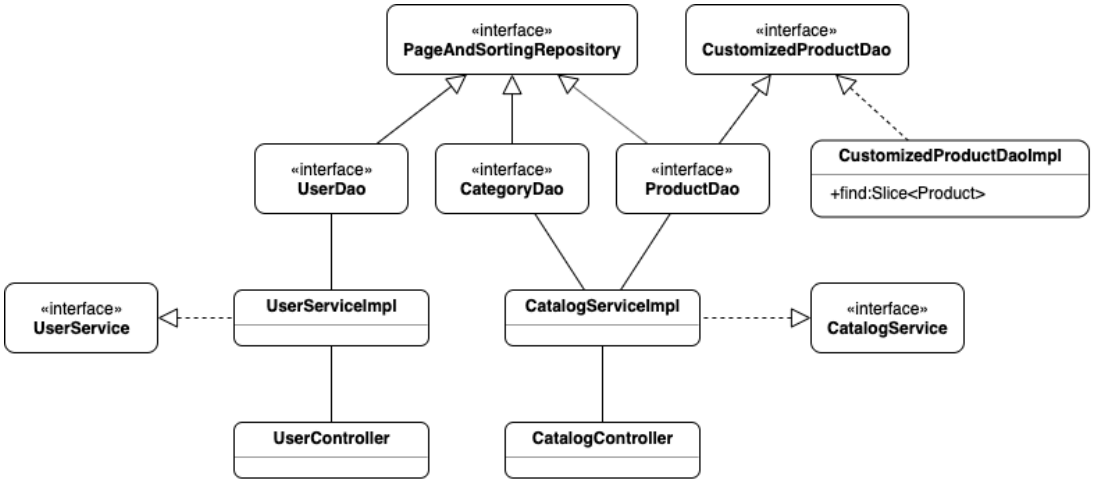


Figura 7.8: Resumen del diseño - Sprint 1

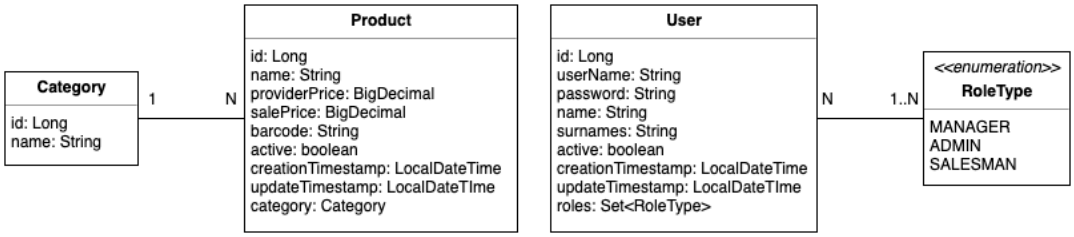


Figura 7.9: Modelo de las entidades del Sprint 1

Dado que los roles de usuario se tratan como un enumerado se puede modelar esta relacion N:M en una simple tabla, a la que nombraremos `UserRole`.

SHSH-24 Autenticación en el sistema

En la implementación de autenticación se decidió emplear el estándar JSON Web Token (*JWT*). El controlador generará un JWT en caso de que el usuario se autentique correctamente en el servicio y en el cual se incorporará la información necesaria para realizar el resto de acciones, como son el identificador del usuario, su nombre de usuario o sus roles. El proceso de autenticación puede verse en la figura 7.10, donde el cliente invoca la acción de autenticación y se encarga de almacenar el token generado para posteriores usos.

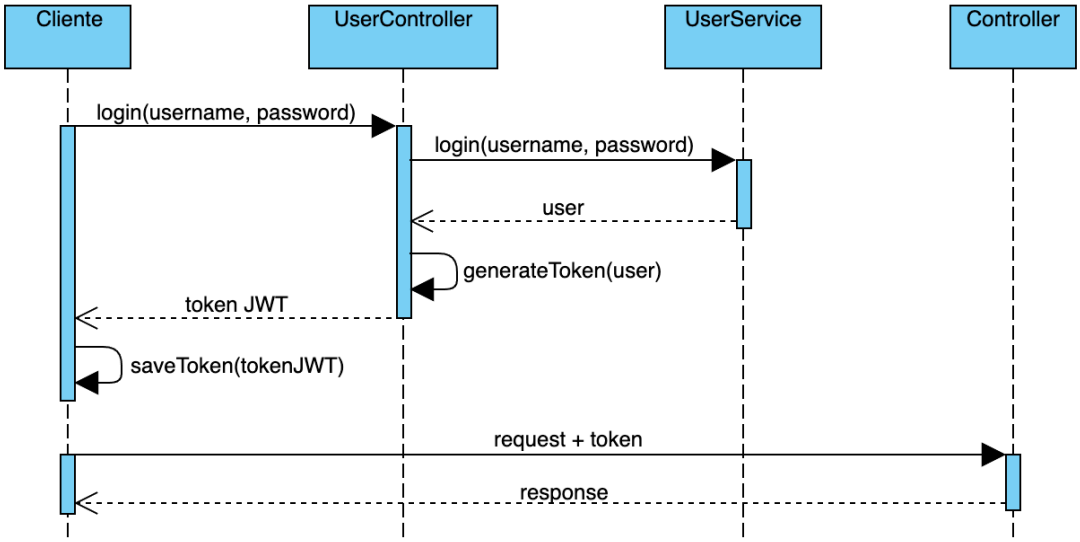


Figura 7.10: Proceso de autenticación

Durante este proceso de autenticación se pueden producir dos casos de error, en los que se generan y controlan las siguientes excepciones:

- `IncorrectLoginException`: Cuando se introduce un nombre de usuario inexistente o unas credenciales incorrectas.
- `UserNotActiveException`: Cuando el usuario introduce unas credenciales válidas pero se encuentra bloqueado en el sistema.

Adicionalmente se incluye una autenticación basada en tokens JWT, a través del cual un usuario recuperará la información necesaria para el uso de la interface enviando el token generado por el servicio. Finalmente destacamos el uso de la *función hash* **bcrypt**, que incorpora Spring Security, durante el almacenamiento de contraseñas y la comparación de estas.

SHSH-5 Agregar categoría

La implementación de esta tarea fue bastante sencilla, mas es la primera tarea que tuvo la condición de que fuese solo accesible por un rol concreto, por lo que hubo que desarrollar un mecanismo para gestionar esta característica compuesto por los dos siguientes factores:

- Renderizado condicional: A través del operador && en la sintaxis JSX de un componente se puede expresar la inclusión condicional de un elemento.
- Método `hasRole`: Método que comprueba si el usuario autenticado tiene alguno de los roles especificados.

```
{hasRole( roles: [Role.ADMIN]) &&
    <Grid item>
        <AddCategory/>
    </Grid>}
```

Figura 7.11: Renderizado condicional en base a roles

SHSH-6 Obtener categorías

Las obtención de las categorías que han sido registradas no tuvo mucha complejidad, pero tienen una gran importancia ya que se relaciona con muchas otras historias de usuario, siendo necesarias a la hora de insertar productos, modificarlos o realizar búsquedas sobre estos. Ya que son un elemento frecuentemente usado se decide realizar la carga de estas categorías tras una correcta autenticación, bien empleando los credenciales del usuario o a través de un JWT previamente almacenado.

SHSH-7 Modificar categoría

La funcionalidad de modificar una categoría es muy similar a la de su creación pero apuntando a una previamente creada, ya que su único atributo es el nombre de la misma. En los casos de modificación de entidades como los que trata esta funcionalidad siempre se controla que desde el servicio no se ejecute sobre recursos inexistentes.

Al completar esta historia de usuario se finaliza la vista de categorías, cuyo resultado puede observarse en la figura 7.12.

SHSH-8 Añadir producto

En el diseño de la entidad que representaría un producto se detalla la necesidad de incorporar marcas de tiempo durante los eventos de creación o modificación de este recurso. Con las anotaciones `@CreationTimestamp` y `@UpdateTimestamp` que proporciona Hibernate se marca aquellos atributos destinados a contener esta información, y con ello la actualización de estos campos se gestionará de manera automática cuando se genere un nuevo producto o este sea modificado.

Adicionalmente se realiza la generación de una cadena que represente al producto, tratándolo como si fuese un código de barras interno para la empresa. Para su generación se emplea la utilidad de Java para generación de identificadores universales **UUID**, y el identificador resultante es codificado posteriormente en Base-32.

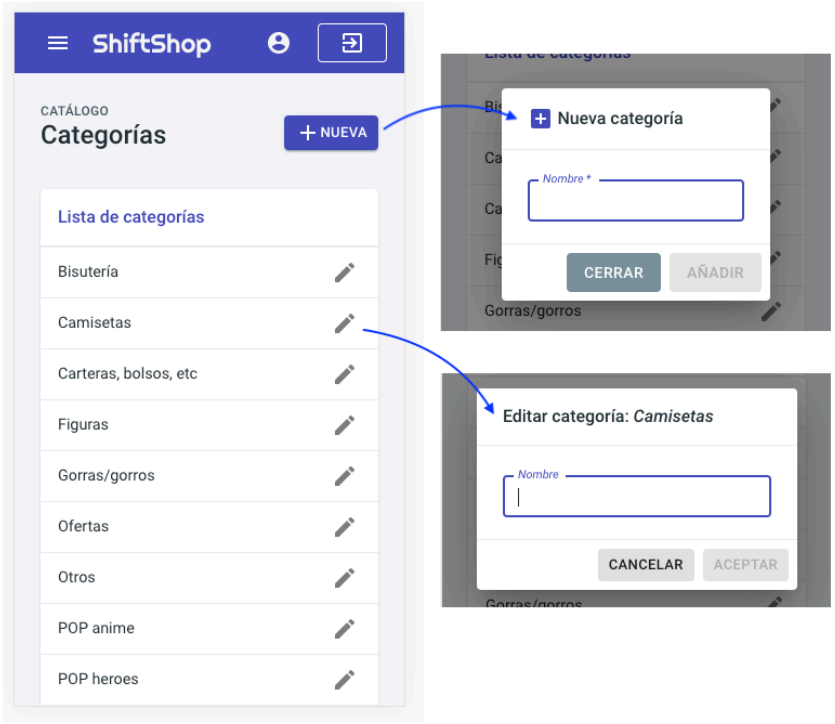


Figura 7.12: Implementación de la vista de categorías

En la elección de la codificación Base32 encontramos los siguientes de ventajas frente a su sucesora, la codificación Base64:

- A pesar de tener una longitud de caracteres mayor vemos beneficioso el uso de solo el espectro de letras en mayúsculas, lo cual se considera de utilidad a la hora de insertar códigos por el usuario. Para la representación usa las 26 letras mayúsculas A-Z y los dígitos del 2 al 7.
- Evita la confusión que producen algunos símbolos durante su uso en impresoras térmicas que tienen unas fuentes muy concretas. Esto se consigue al omitir el uso del 1 y el 0, ya que podrían ser interpretados por las vocales mayúsculas *I* u *O*.

En esta implementación se diseñó el componente de selección de categorías `CategorySelector` y, dado que será un elemento común a varias funcionalidades como la búsqueda de productos o la modificación de estos, se diseñó específicamente para que fuese reutilizable gracias a las dos propiedades siguientes:

- `empty`: Añade la opción vacía al selector para casos de inserción.
- `allCategories`: Añade una opción al selector que representa la selección de todas las categorías, útil para casos de búsqueda.

Finalmente, el usuario es quien completa el resto de atributos necesarios para la creación del producto: nombre, categoría, precio del proveedor y precio de venta.

SHSH-10 Búsqueda sobre productos

La búsqueda de productos se realiza en base a dos parámetros principalmente, el primero será la categoría de los productos y el segundo una serie de palabras clave. Esta búsqueda siempre será con paginación y tendrá un diseño adaptado al tamaño de pantalla del dispositivo, mostrando los resultados en una tabla para los dispositivos con pantallas más amplias mientras que para dispositivos tipo móvil se mostrarán los resultados en una lista de tarjetas. Ambas representaciones pueden observarse en la figura 7.13.

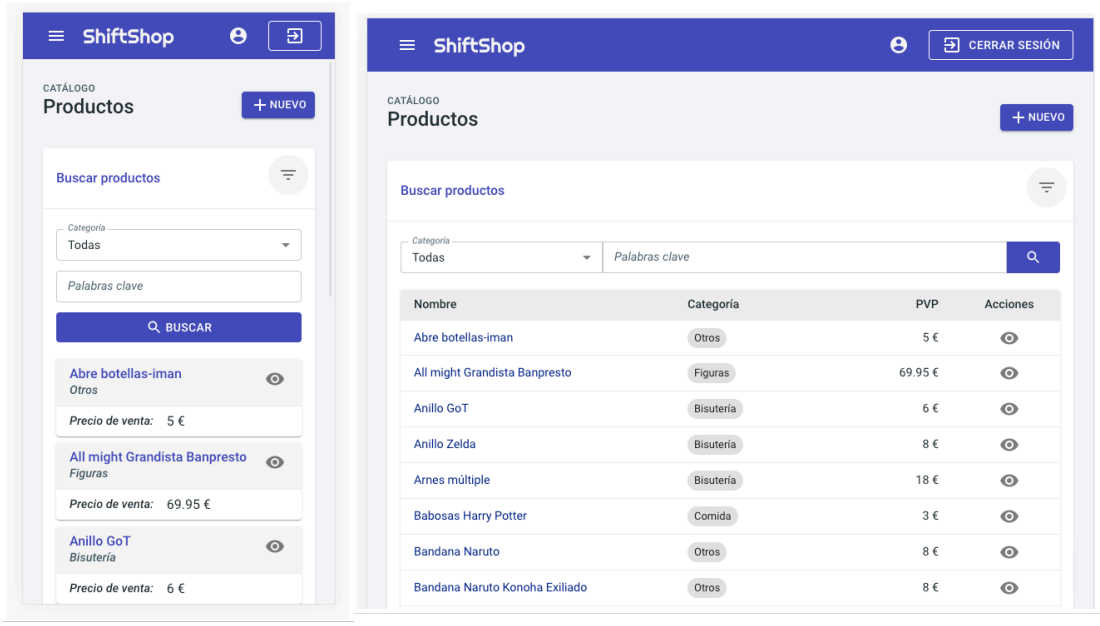


Figura 7.13: Diseño adaptativo en la búsqueda de productos

SHSH-9 Obtener detalle de producto

Un punto clave a la hora de mostrar los detalles de un producto es la información que se devuelve para cada rol. La solución diseñada para esto incluye una pequeña clase con la lógica que permite:

- Obtener los roles del usuario que invoca un método de algún controlador. Para esto se incluye el argumento `Authentication` en el método del controlador, y cuyo valor será resuelto por el framework Spring.

- Comprobar si, dados unos roles especificados, entre ellos se encuentra alguno de los roles definidos en el usuario de la petición. Este método se usará durante la conversión Entidad-DTO en aquellos atributos que solo quieran ser visibles para ciertos roles.

La diferencia en la vista del detalle de un producto para el personal de venta se encuentra en que este no recibe información sobre el precio de compra del producto, por lo que tampoco se le mostrará el beneficio y el ROI calculado. Esto puede observarse en la figura 7.14.

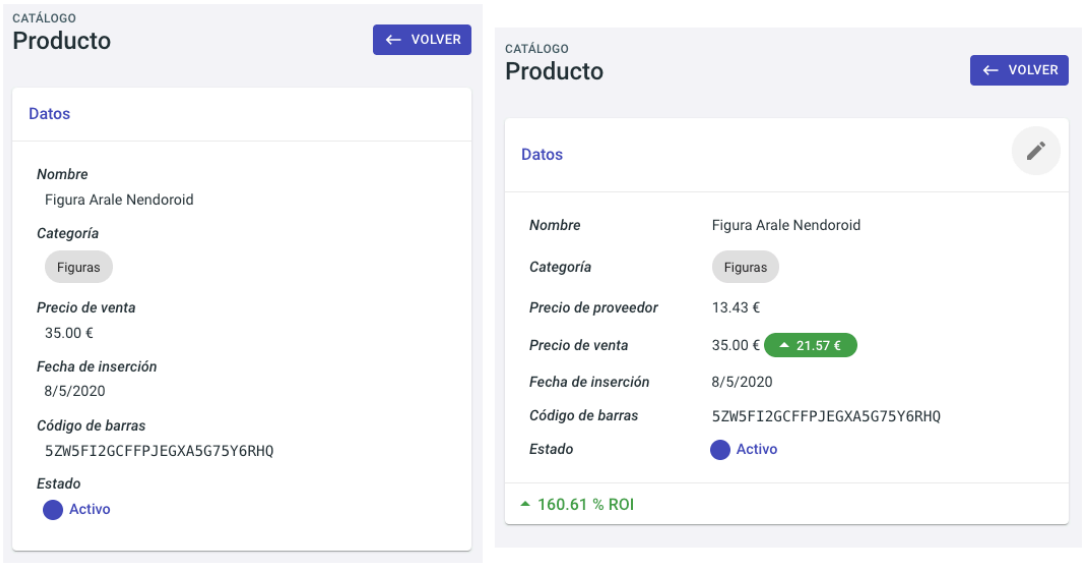


Figura 7.14: Vista de un producto para personal de venta vs. administrador/gerente

SHSH-11 Modificar datos de producto

La modificación de un producto se añade a la vista del mismo producto. El administrador podrá modificar los mismos atributos que completaba durante la creación, permitiendo editar a mayores el código de barras que les fue otorgado ya que puede darse el caso de productos que dispongan de un código propio que siga unos estándares globales.

Con esta historia de usuario se pudo comprobar la actualización de las fechas de modificación registradas en los productos, esencial para próximas funcionalidades.

Tareas adicionales

En adición a las historias de usuario desarrolladas se realizó una serie de tareas que se describen a continuación:

- Creación del modelo de BBDD: Se generó la primera versión del modelo de la BBDD que se utilizaría durante el desarrollo de las historias de usuario. Además se creó un *script*

específico para producción, en el que se generaría de manera automática el usuario con el rol Gerente y Administrador que controlaría la primera versión del sistema.

- Despliegue del sistema: En este sprint se generó el documento `INSTALL.md`, que contiene una guía para la correcta instalación en el equipo destino de las herramientas necesarias para el funcionamiento del sistema. Durante esta tarea se investigó:
  - Despliegue seguro de aplicaciones Spring Boot y ejecución durante el arranque del servidor [49].
  - Uso del servidor de contenido estático **nginx** [50], que actuará a su vez como un proxy inverso para redireccionar las peticiones que se lancen desde la aplicación SPA al servicio.
  - Creación de certificados internos para securizar las conexiones con el servidor nginx.

7.3 Sprint 2

7.3.1 Análisis

En este Sprint se definió como objetivo el **registro de ventas durante eventos**, por lo que sería necesario también comenzar a **registrar personal en el sistema**. Siguiendo con la dinámica vista en el Sprint anterior, se crearon los prototipos y un diagrama de casos de uso (7.15) que refleja la funcionalidad del punto de venta.

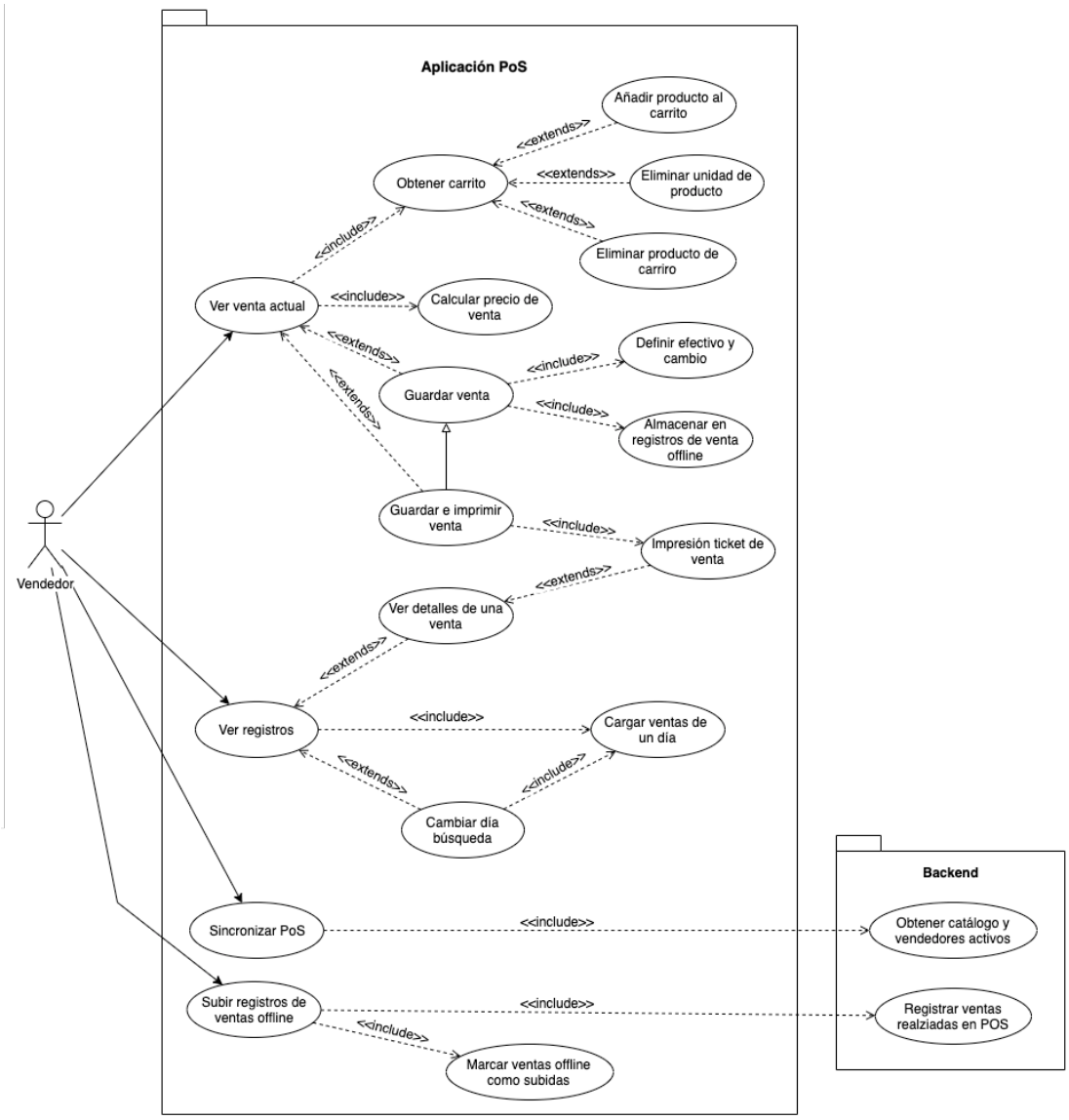


Figura 7.15: Casos de uso del punto de venta

7.3.2 Diseño e implementación

En el diseño de la funcionalidad destinada a la gestión de usuarios se investigó la opción de dividir el paquete que se genera con la interface de usuario, usando la **fragmentación basada en rutas** que expone la página de React [51]. Con esto no solo se mejora los tiempos de carga iniciales al reducir el tamaño del paquete da descargar, sino que evita que los usuarios que no tengan acceso a ciertas vistas no se descarguen ese trozo de funcionalidad.

La figura 7.16 muestra los principales cambios incluidos para dar soporte a la funcionalidad de este sprint, incluyendo el nuevo controlador que se invocará durante el proceso de sincronización de cada punto de venta.

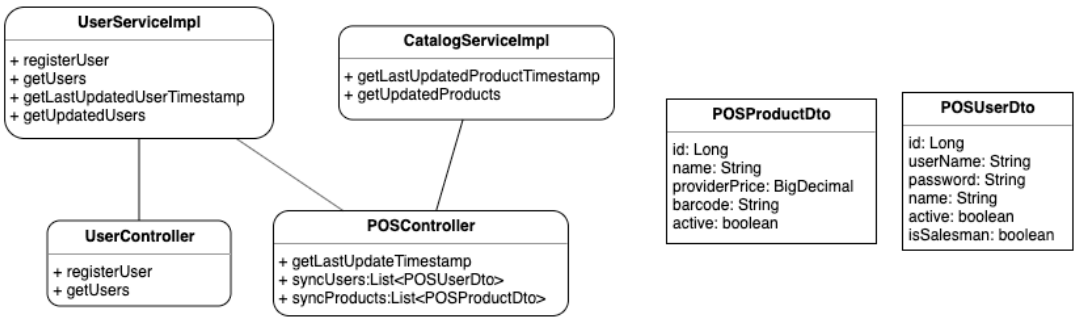


Figura 7.16: Resumen del diseño - Sprint 2

SHSH-12 Activar y desactivar producto

Con esta sencilla historia de usuario se aborda la funcionalidad para deshabilitar aquellos productos que ya no estén disponibles, o habilitarlos en el caso menos frecuente de que se vuelvan a ofrecer. Su beneficio principal es reducir el catálogo visible al personal de venta y facilitar el proceso de búsqueda de productos en el terminal PoS durante una venta.

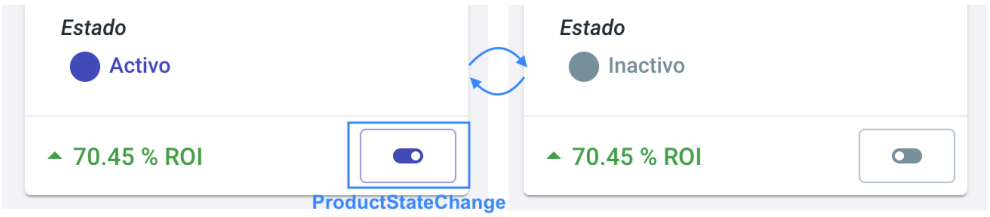


Figura 7.17: Transición entre los estados del producto

SHSH-15 Registrar personal

Hasta esta historia de usuario se contaba con el usuario con el rol gerente y administrador autogenerado por el sistema como único usuario del sistema. Esto cambió gracias a esta funcionalidad, ya que sería necesario dar de alta a los empleados que trabajarán con la aplicación PoS durante los eventos.

En la inserción de usuarios el servicio eliminará el rol gerente en caso de ser incluido en la petición, manteniendo así la lógica de un **único usuario con la funcionalidad de gestión sobre el personal**. Tras esto se comprobará además que entre los datos de usuario se elige un nombre de acceso único y que tiene al menos un rol asignado.

Previendo la necesidad de un selector múltiple de roles en más de alguna funcionalidad se diseñó un componente reusable para este propósito (`RoleMultiSelector`), otorgándole un color característico a cada rol para mejorar la experiencia de usuario. A este componente se le podrá incluir una lista con los roles que no se podrán seleccionar, útil a la hora de insertar nuevos usuarios para el personal, ya que podremos omitir la inclusión del rol gerente. Tanto el componente mencionado como el diálogo de creación de usuarios resultante puede verse en la figura 7.18.

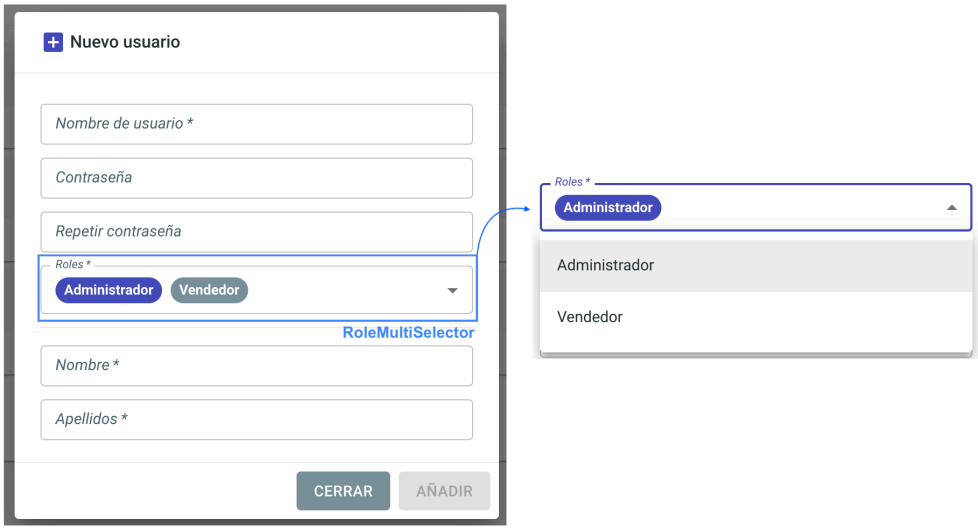


Figura 7.18: Diálogo de creación de nuevos usuarios

SHSH-16 Listar personal registrado

En la vista del personal se mostrará los usuarios que están en activo en primera instancia, ya que se considera el principal grupo sobre el que se querrá actuar. Se ofrece a mayores, completando así la visualización de usuarios, la opción de incluir en la visualización aquellos usuarios que fueron bloqueados.

Siguiendo el modelo adaptativo visto en la búsqueda de productos se expondrá la información en forma de tarjetas para pantallas con dimensiones propias de dispositivos móviles, mientras que en dimensiones más amplias se visualizará una tabla con el mismo contenido. Esto puede verse en la figura 7.19.

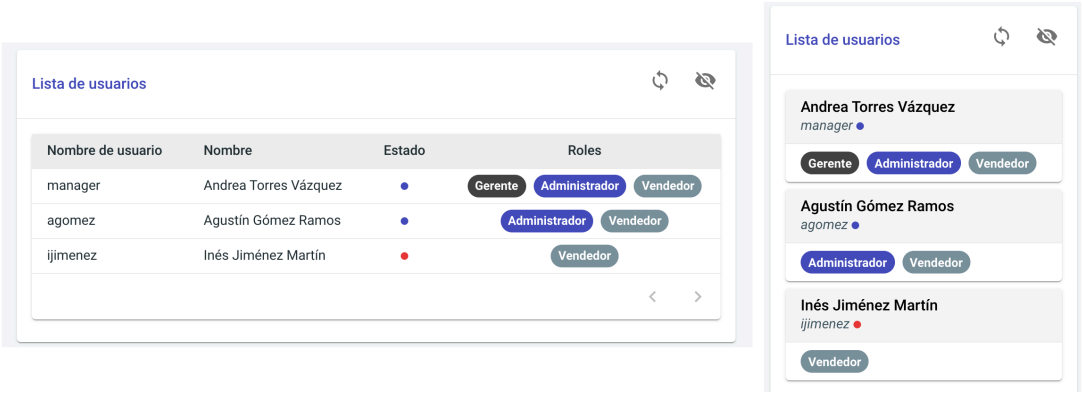


Figura 7.19: Vista de usuarios

Al contener toda la información sobre un usuario en esta vista no es necesario implementar otra destinada exclusivamente a los detalles de estos.

SHSH-20 Actualizar catálogo y usuarios en PoS

La funcionalidad para actualizar los datos del PoS dependerá de dos elementos:

- **Fecha de última actualización**: El servicio ofrece un método que devuelve la fecha de modificación más reciente entre productos y usuarios, y que será usado por el terminal a la hora de comprobar si existe alguna actualización disponible en los datos. Esta fecha será visible en la página inicial de la aplicación para información del usuario como se observa en la figura .
- **Proceso de sincronización**: En caso de solicitar una sincronización se ejecuta el proceso destinado a esta tareas, y que se compone de tres pasos:

  1. **Autenticación en el servicio**: con lo que se obtendrá el JWT que autorizará la descarga posterior de datos.
  2. **Descarga de usuarios y productos**: El servicio expone dos métodos, uno para cada tipo de elemento a sincronizar. Ambos métodos reciben como parámetro una marca de tiempo y todo aquel elemento cuya fecha de modificación sea posterior a la marca especificada se incluirá en los datos a sincronizar.

     Estos datos se enviaran en un formato reducido, los cuales contendrán solamente aquella información necesaria durante el registro de ventas. Adicionalmente se

incluye el estado de estos elementos, con lo que se podrá conocer si un elemento debe ser almacenado/modificado o bien eliminado de los registros locales.

3. **Completar proceso**: Una vez actualizados los datos se almacenará la fecha de última modificación obtenida, la cual será empleada en la siguiente sincronización.

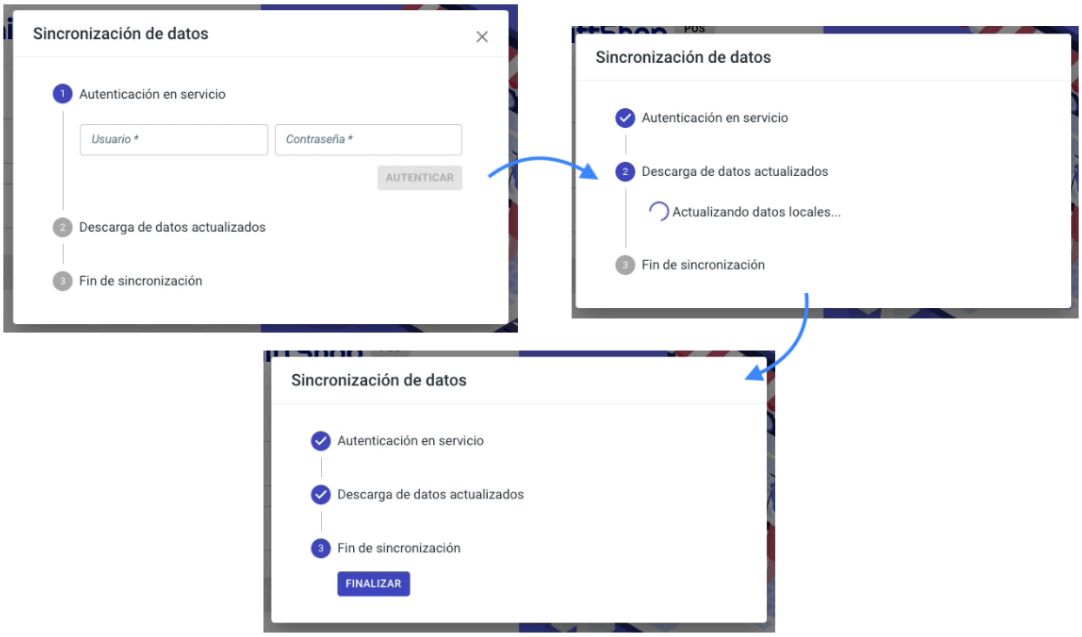


Figura 7.20: Proceso de sincronización en PoS

Con estos elementos se pretende realizar un control eficaz sobre los datos a actualizar, ya que se descargan solo aquellos que han sufrido una o más modificaciones. La comprobación de actualizaciones disponibles se produce durante la carga de la pantalla inicial de autenticación, evento que ocurre al arrancar la aplicación o cerrar la sesión por parte del usuario.

SHSH-21 Identificación en PoS

La identificación en los terminales se realizará contra la versión de los datos del personal de venta almacenamos localmente por los terminales. Ya que sus contraseñas de acceso se guardan *hasheadas*, al igual que en el servicio, se necesita una utilidad que verifique la correcta correspondencia durante el proceso de autenticación. La librería bycriptjs [52] suple esta necesidad, por lo que fue incorporada al proyecto.

SHSH-18 Registrar venta en PoS

Esta historia de usuario se fragmentó en la implementación de tres tareas diferentes ya que se consideró que poseía una complejidad mayor a las realizadas hasta la fecha.

La primera de las tareas estaría enfocada en el **control del carrito de venta y agregar elementos** al mismo. Para ello se incorporó al estado Redux el elemento que representará el carrito, una lista con el identificador de cada producto y su cantidad en el carrito, además de las siguientes acciones:

- ADD_TO_CART: Incorpora un producto determinado al carrito en caso de no estar presente todavía, o lo incrementa en una unidad en caso contrario.
- SUBTRACT_PRODUCT_IN_CART: Sustrae una unidad de un producto presente en el carrito, eliminándolo de este en caso de que se sustraigan todas las unidades.
- REMOVE_FROM_CART: Elimina del carrito un producto junto con todas sus unidades.
- CLEAR_CART: Limpia por completo de productos el carrito.

Para agregar estos productos se implementó un componente de selección con el catálogo local cargado (7.21), a través del cual el vendedor buscará y agregará elementos al carrito.

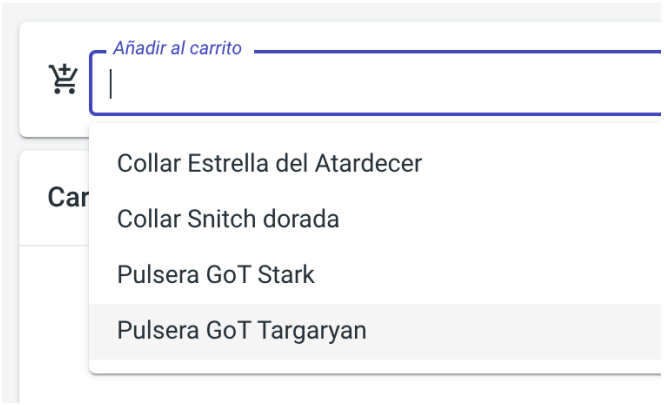


Figura 7.21: Componente para agregar productos al carrito

En la segunda tarea se incorporó la visualización del carrito y las posibles acciones sobre el mismo, enlazando así con la funcionalidad desarrollada en la tarea previa. Además se incluyó dos utilidades que se consideraron interesantes para el usuario:

- Componente `CartBadgeCount`: mostrará los elementos totales del carrito y será visible en todas las vistas de la aplicación.
- Durante el cierre de sesión se alertará en un dialogo en caso de que existan elementos pendientes en el carrito.

El resultado de esta tarea puede verse en la figura 7.22.

Figura 7.22: Carrito de compra

Por último, la tercera tarea se centra en completar la venta y almacenarla localmente. Como se describe en el análisis de esta historia de usuario se incorporó la posibilidad de agregar un descuento sobre la venta total y un cálculo sobre el cambio a entregar a los clientes. Estos dos valores tendrán un límite, controlando que el descuento no supere el importe total y que el efectivo indicado no sea inferior al importe final.

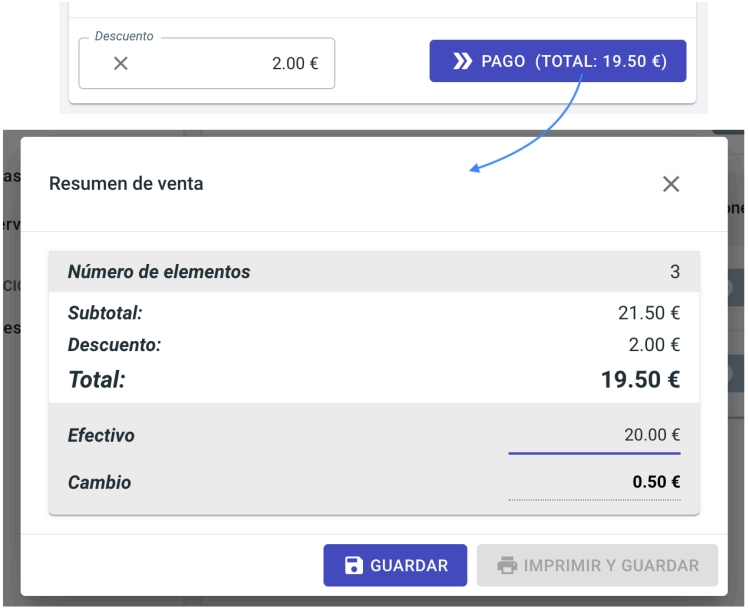


Figura 7.23: Finalizar venta en PoS

Durante el registro de la venta se tuvo en consideración dos aspectos:

- **Generación de un identificador para las ventas**: El cliente recibirá un ticket sobre su venta, por lo que será necesario generar algo que las identifique sobre el resto unívocamente. Para ello se recurre al generador de identificadores universales de la librería uuid [53]. Este identificador solucionará además posible errores de conexión durante la sincronización de las ventas, ya que al contar con un identificador universal podremos emplear la petición idempotente PUT.
- **Almacenamiento local de las ventas con su fecha como clave**: Una de las recomendaciones sobre PouchDB es el uso de claves primarias sobre atributos que serán muy utilizados durante la búsqueda de documentos. Dado que la fecha de una venta será única en cada terminal PoS y será el principal atributo sobre el que se realicen búsquedas, se decidió usar este atributo usando el estándar ISO-8601 como clave primaria de estos documentos.

  Un ejemplo de este funcionamiento sería la búsqueda sobre un día concreto, por lo que para el día 9 de agosto de 2020 se recogerán aquellos documentos cuyo identificado empiece por la cadena `"2020-08-09"`.

SHSH-22 Búsqueda de ventas registrada en PoS

La aplicación PoS contará con una vista destinada a mostrar las ventas que se han registrado durante los eventos. Tendrá un selector de fecha que cargará el día actual y sus ventas por defecto. Para la selección del día la librería Material-UI Pickers [54] dispone de una serie de componentes destinados a este aspecto.

Dado que al finalizar el día es importante realizar un cuadre de las cajas se aprovecha la carga de las ventas del día seleccionado para incluir el cálculo de la facturación registrada en el día seleccionado (figura 7.24).

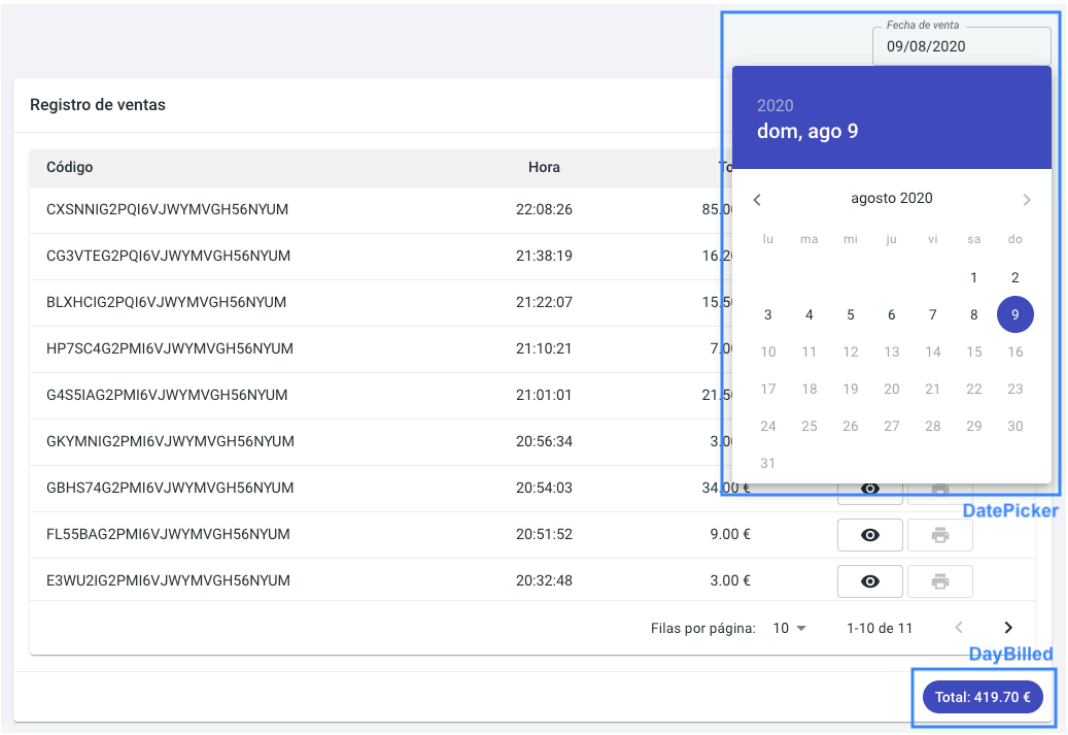


Figura 7.24: Vista de registros locales en el punto de venta

Desde el listado cargado se podrá seleccionar una venta para visualizar sus detalles, tal como puede observarse en la siguiente figura (7.25).

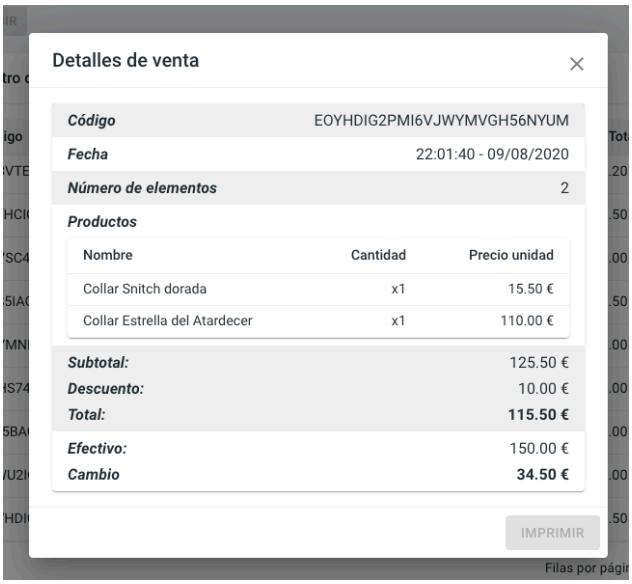


Figura 7.25: Detalles de venta en el punto de venta

7.4 Sprint 3

7.4.1 Análisis

Se marca como objetivo final del proyecto la **visualización y resumen sobre las ventas realizadas**. Además, para finalizar con las historias de usuario fijadas, se completará la actualización de usuarios del personal y la impresión de tickets en la aplicación PoS.

7.4.2 Diseño e implementación

En la figura 7.26 se resume el diseño de los nuevos métodos expuestos por el servicio y las clases a implementar.

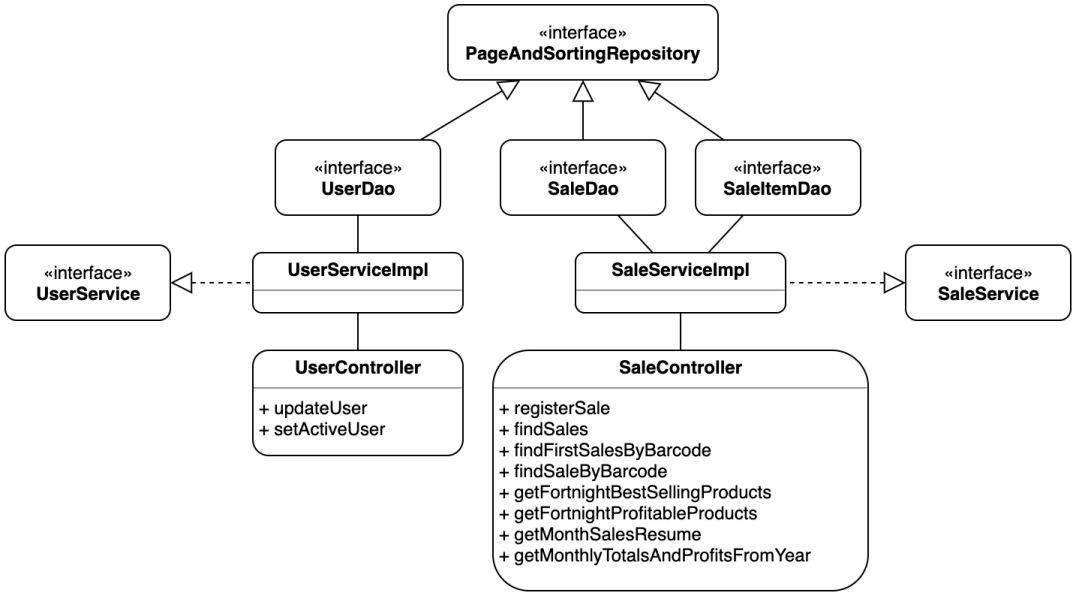


Figura 7.26: Resumen del diseño - Sprint 3

El modelo de la base de datos agregará las siguientes entidades y relaciones:

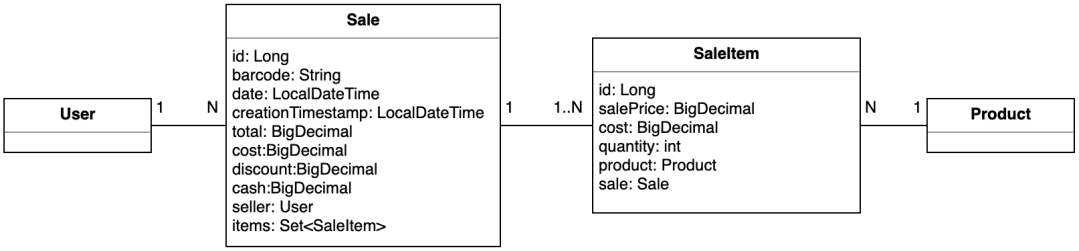


Figura 7.27: Modelo de entidades incorporadas en el Sprint 3

Se puede observar la *desnormalización* en la tabla de ventas al incluir los atributo derivados `total` y `cost`, los cuales podrían ser calculado a través de los respectivos importes de cada elemento incluido en la venta. La decisión de incluir esta desnormalización se basa en la eficiencia que se consigue durante la visualización de las ventas, eliminando el proceso de cálculo sobre cada una de las ventas a devolver. Nos beneficiamos además de una característica de las ventas, ya que una vez realizadas no podrán ser modificadas, evitando así la problemática de controlar la posible variación en el valor de estos atributos derivados.

SHSH-17 Actualizar datos del personal

Con esta historia de usuario se permitirá al gerente modificar los datos de cada empleado salvo el nombre de usuario y la contraseña de autenticación, la cual podrá ser modificar solo por su propio usuario. También se podrá modificar los roles asignados siguiendo las mismas restricciones vistas durante la inserción de estos.

Además se incluirá en esta funcionalidad la opción de bloquear y desbloquear las cuentas de usuario, a excepción de la destinada al gerente del sistema por motivos obvios.

A la visualización de empleados se agregó un apartado de acciones, desde la que se podrá invocar el dialogo de edición o bloquear/desbloquear directamente al usuario (figura 7.28).

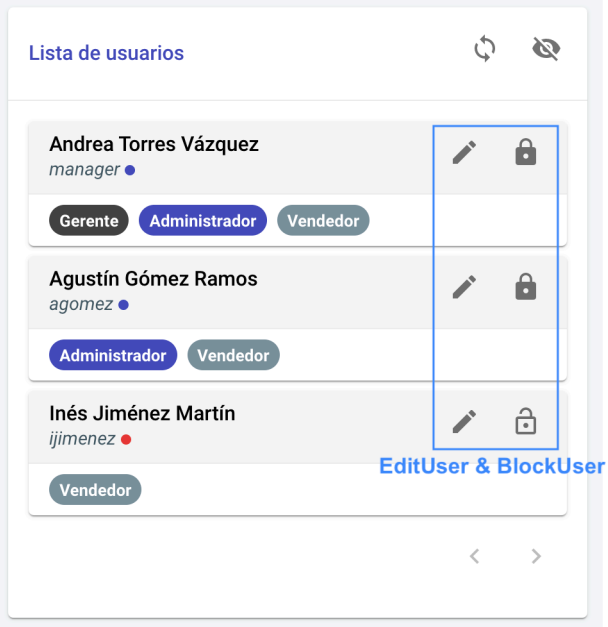


Figura 7.28: Nuevas acciones disponibles sobre usuarios

SHSH-23 Registrar ventas de PoS en servicio

El proceso de sincronizar las ventas en el servicio se lanzará desde la vista de registros locales de la aplicación PoS, en la que se dispone de un botón para iniciar este proceso que además mostrará el número de ventas pendientes por sincronizar.

El proceso se fragmenta en una serie de pasos, similar al visto durante la sincronización de los datos del terminal, y que de hecho reutilizará el componente de autenticación para proceder con la subida de ventas. Durante el paso de sincronización se puede diferenciar tres posibles escenarios:

- El *happy path* es el escenario donde todo se resuelve de manera satisfactoria y sin errores durante la sincronización.
- Error durante la sincronización de alguna venta al intentar almacenarla en el servicio, tras lo que se devolverá información sobre las causas del mismo. El proceso de sincronización continuará aunque ocurra alguno de estos errores.
- Caída del servicio, situación en la que se opta por finalizar el proceso de sincronización.

Tras la ejecución del proceso, en cualquiera de los escenarios vistos, se ofrece un resumen con el número de ventas sincronizadas y un informe con los diferentes errores capturados.

Se decidió que el proceso de subida de ventas se realizaría en una serie de rondas de 100 elementos que se lanzarán de forma asíncrona, utilizando para ello la funcionalidad `Promise.all` que notificará cuando todos los elementos de una ronda hayan sido tratados (figura 7.29). Esto aprovechará las características asíncronas del sistema junto con un control sobre el consumo de los recursos del servicio por parte de los terminales con la aplicación PoS. Como punto extra se permitirá pausar la sincronización, saltando las siguientes rondas pendientes y mostrando el resumen de sincronización final.

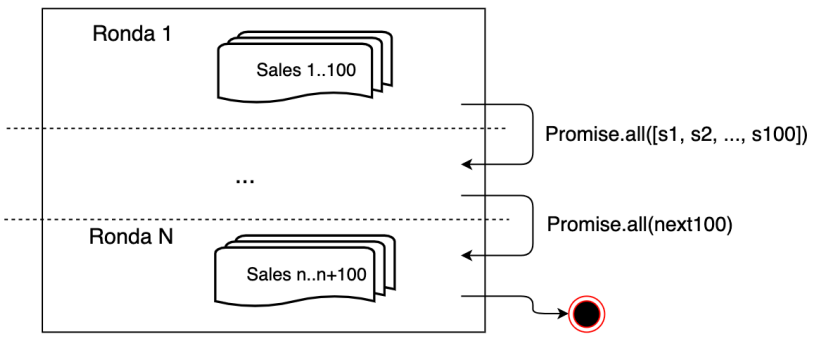


Figura 7.29: Sincronización de ventas por rondas

Durante la persistencia de las ventas en el servicio se incluirán los costes a cada elemento de la venta, con lo que se determina el beneficio aproximado obtenido.

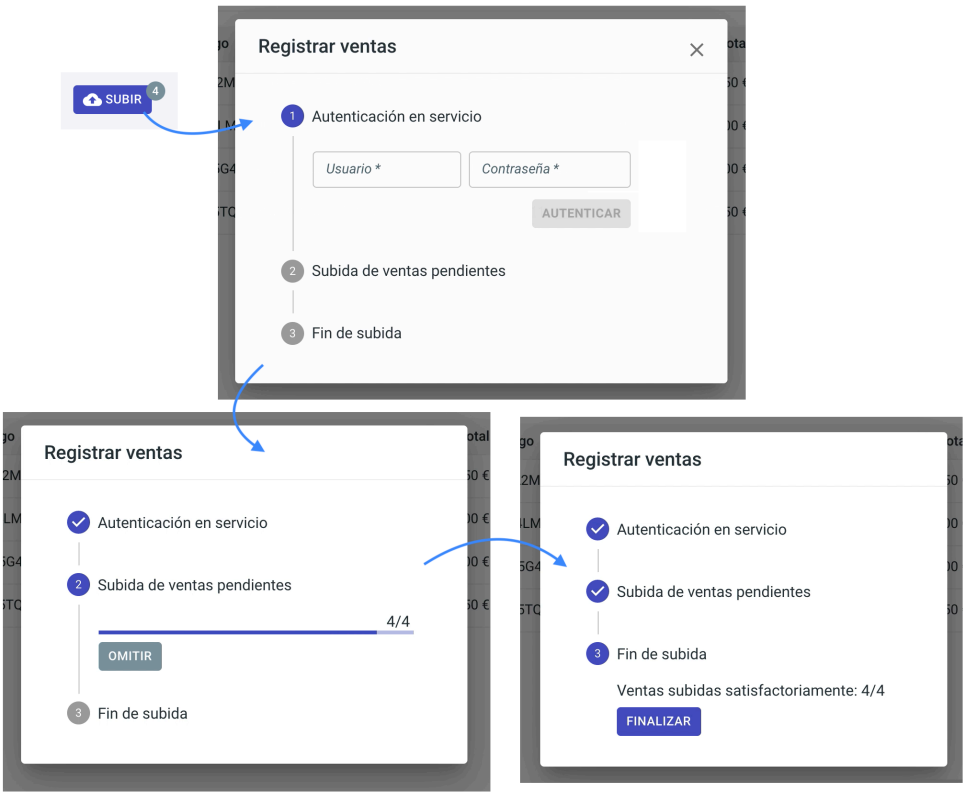


Figura 7.30: Proceso de sincronización de ventas en servicio

SHSH-13 Búsqueda sobre ventas realizadas

La búsqueda de ventas tendrá una vista en el que se podrá seleccionar un rango de fechas sobre el que recopilar información de ventas, controlando siempre que este rango sea válido. Contará además con múltiples factores de ordenación sobre los detalles más relevantes, tales como la fecha, el importe total o el código de barras que las identifica. Por defecto se marcará un rango inicial de 30 días, previos al instante actual, y ordenando por su fecha de forma descendente, ya que se considera el factor de ordenación más frecuente en estas búsquedas.

Se dio uso del diseño adaptativo visto en las funcionalidades que listaban productos y usuarios, ajustando la representación de la información dependiendo del tamaño de pantalla (figura 7.31).

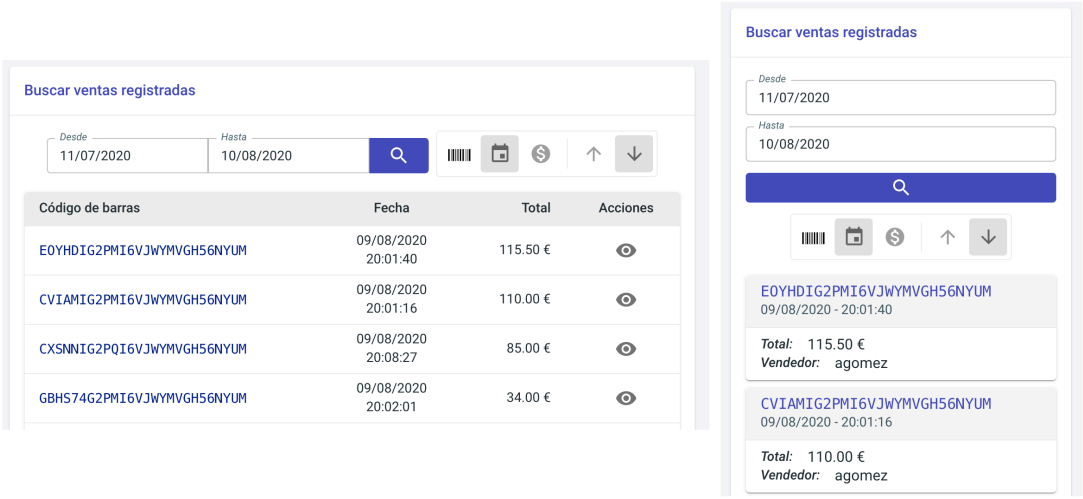


Figura 7.31: Vista de las ventas registradas en el servicio

SHSH-27 Obtener detalles de venta

La funcionalidad de esta historia de usuario permitirá acceder a los detalles de una venta concreta desde los resultados de una búsqueda (SHSH-13), resultado en la respectiva vista de la venta (figura 7.32).

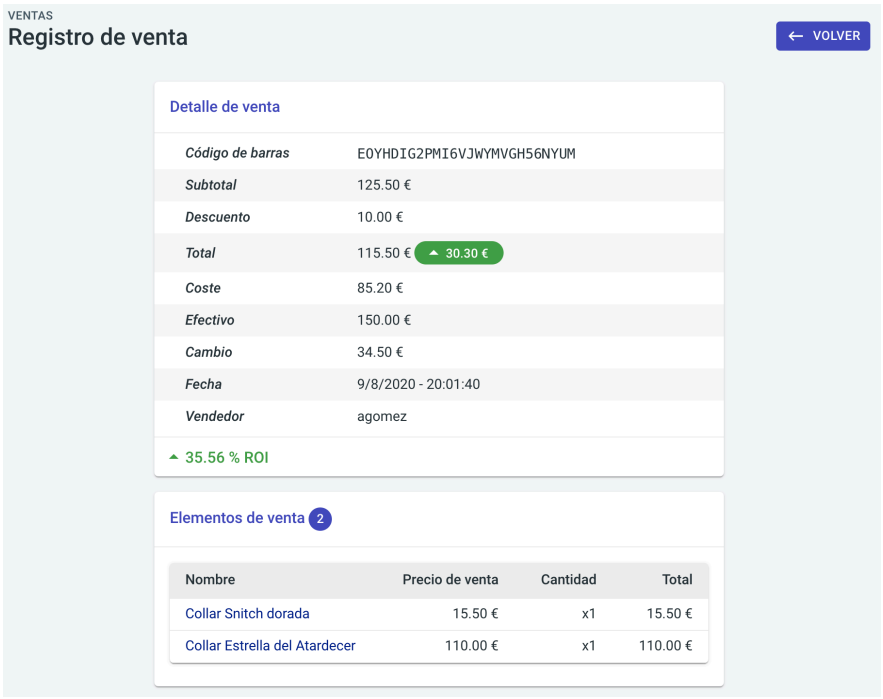


Figura 7.32: Vista de los detalles de una venta

Adicionalmente se incluyó la utilidad de un buscador de ventas en base a su código, útil cuando se quiera buscar por esta referencia que fue entregada al cliente. Se destina para esto un cuadro de búsqueda que, para mejorar la experiencia de usuario, ofrecerá el **autocompletado** de los posibles códigos buscados (7.33).

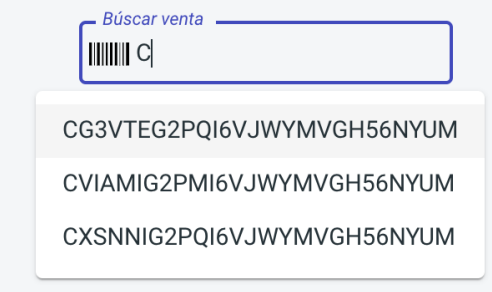


Figura 7.33: Componente `BarcodeAutocomplete`, buscador de códigos de ventas

Durante la introducción de caracteres por parte del usuario se podría ocasionar muchas llamadas sobre la búsqueda de códigos de venta al servicio, llamadas que podrían omitirse ya que la última será la que se muestre al usuario finalmente y la que tenga información más relevante. Por este motivo se indagó sobre algún mecanismo que limitase el número de llamadas que se invocan mientras el usuario introduce caracteres en este componente, encontrando así las técnicas *debouncing* y *throttling*. La técnica *throttling* consiste en limitar en un número máximo las llamadas que se producen en un intervalo de tiempo, mientras que la técnica *debouncing* permite agrupar múltiples llamadas secuenciales en una única, siempre que la distancia en el tiempo entre ellas no sea superior al del intervalo definido (7.34).

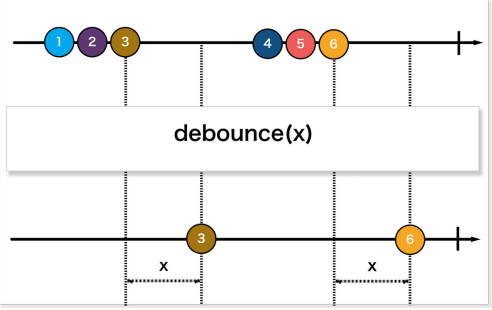


Figura 7.34: Técnica debouncing

La técnica *debouncing* será finalmente la seleccionada a incorporar, ya que es la que más se ajusta al efecto que queremos, permitiendo al usuario introducir múltiples caracteres rápidamente e invocando a la funcionalidad cuando se detenga la introducción de caracteres.

SHSH-14 Obtener resúmenes sobre ventas

Esta historia de usuario está destinada a otorgar una valor extra al registro de las ventas mostrando unos resúmenes sobre estas, con las que gerente y administradores puedan realizar un seguimiento y análisis.

Tal como se mencionó su análisis, se ofrecerán unos resúmenes de ventas sobre los últimos 15 y 30 días (omitiendo el día actual) y resumen anual, siendo la página principal de la web SPA la que exponga toda esta información a modo de *dashboard*. Puesto que se tienen en cuenta solo los registros hasta el día anterior la variación diaria de estos resúmenes es nula, por tanto se podrá mejorar la eficiencia en la visualización de los mismos al solicitarlos y almacenarlos en el estado de la aplicación una única vez durante el acceso a esta.

Dado que los datos a mostrar son calculados en base al conjunto de ventas de un determinado periodo, se vio la necesidad de incluir alguna clase en el servicio cuya representación englobase este tipo de datos. Indagando entre la documentación del proyecto Spring Data se encontró una utilidad con la que recoger estos datos agregados a través de las consulta al DAO de esta entidad, las proyecciones [55], definidas como clase o interface. Un ejemplo de su definición se muestra en la figura 7.35, mientras que su uso puede observarse en la figura 7.36.

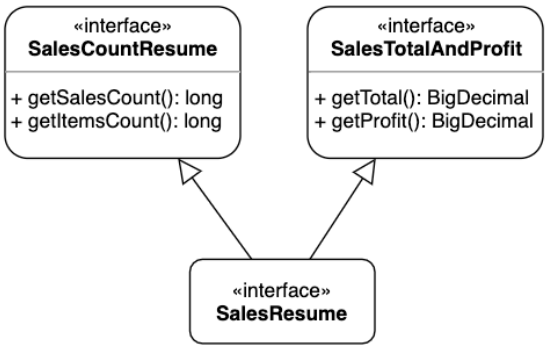


Figura 7.35: Representación de las proyecciones definidas

```
@Query("SELECT COUNT(distinct s.id) AS salesCount, COALESCE(SUM(si.quantity), 0) AS itemsCount " +
        "FROM Sale s JOIN SaleItem si ON s.id = si.sale.id AND s.date BETWEEN ?1 AND ?2")
SalesCountResume getSalesCountSummary(LocalDateTime initDate, LocalDateTime endDate);

@Query("SELECT COALESCE(SUM(s.total), 0) AS total, COALESCE(SUM(s.total - s.cost), 0) AS profit " +
        "FROM Sale s WHERE s.date BETWEEN ?1 AND ?2")
SalesTotalAndProfit getSalesTotalAndProfit(LocalDateTime initDate, LocalDateTime endDate);
```

Figura 7.36: Consulta con funciones de agregación sobre `SaleDAO` empleando las proyecciones

En la figura 7.38 se muestra el panel con el resumen de los últimos 15 y 30 días.

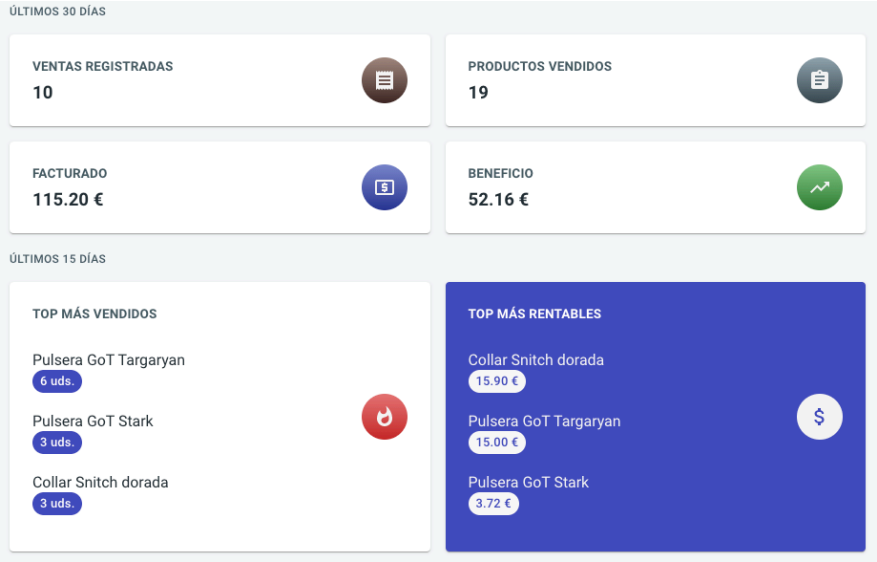


Figura 7.37: Panel con el resumen de los últimos 15 y 30 días

Por otra parte, la desnormalización que fue aplicada sobre las ventas incrementará la eficiencia en la obtención de la facturación y costes del resumen anual, ya que se omite la consulta y cálculo sobre los elementos de cada venta. La visualización de este resumen mostrará el rendimiento mes a mes de los beneficios obtenidos frente a la facturación (figura 7.38), usando para ello las librerías de representación de gráficas `chart.js` y `react-chartjs-2`.

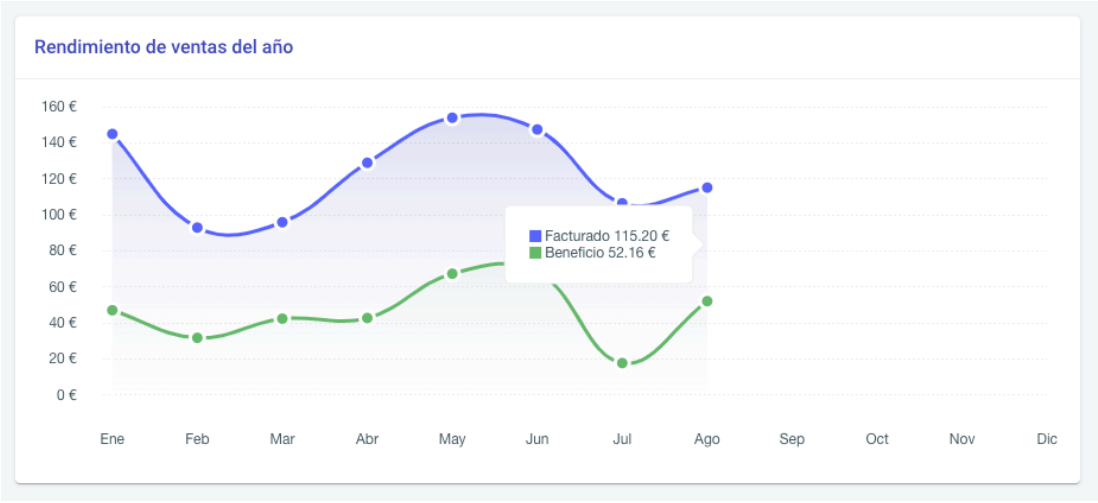


Figura 7.38: Panel con el resumen de los últimos 15 y 30 días

SHSH-19 Impresión de ticket de venta

En esta funcionalidad se dispondrá de un kit de desarrollo software (SDK) que ofrece el vendedor de la impresora que usaremos, que contendrá un manual con la guía de configuración y conexión con la impresora, una API con todos los comandos disponibles a utilizar y la librería JavaScript que ofrecen para consumir estos comandos. La figura 7.39 muestra los pasos a implementar en la aplicación para la conexión y uso de la impresora.

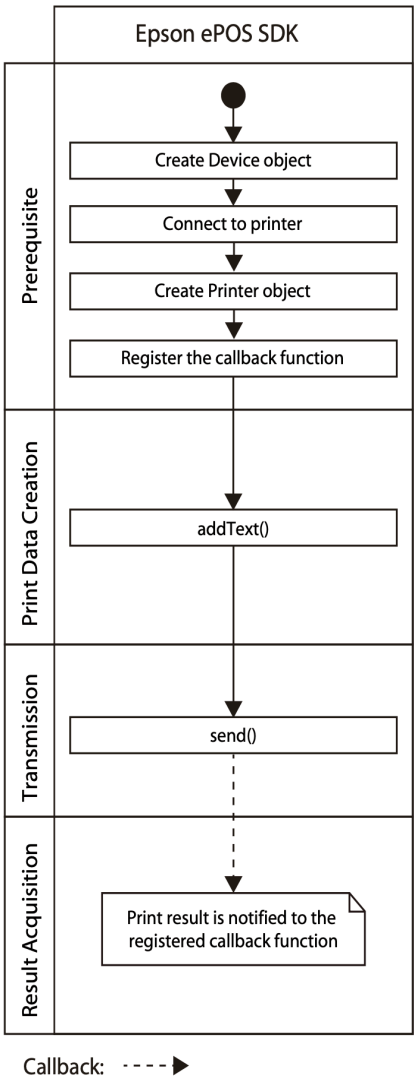


Figura 7.39: Proceso de conexión y uso de impresora a través del SDK

Para agregar la librería del SDK a nuestra interface se emplea el componente Script del paquete npm `react-load-script`, en el que se indicará la ruta local donde se almacena esta librería y dos funciones que se ejecutaran en caso de éxito o error respectivamente.

La implementación se basó en una demo que contenía el SDK, agilizando así bastante la integración. Por otra parte se incluyo un control sobre el estado de la conexión (CONNECTED, DISCONNECTED y CONNECTING) y una transición entre estos estados durante el proceso visto, que permitirá mostrar la disponibilidad de uso de la impresora a los usuarios y bloquear la ejecución de las tareas de impresión cuando esta no esté accesible.

Dado que la conexión con esta impresora se basa en dos atributos variables (dirección IP y nombre de dispositivo) que pueden ser modificados con las herramientas de configuración que proporciona el vendedor, se vio la necesidad incluir en la aplicación estos valores editables e incluir una vista para su edición, donde además se ofrecerá la funcionalidad de conexión/reconexión en caso de fallo durante la comunicación con el dispositivo (figura 7.40).

Figura 7.40: Configuración de impresora y descripción de sus componentes

Finalmente se completó el mensaje que iría en la petición `send` del API de la impresora, con todos los datos respectivos de la venta a ofrecer al cliente, y que resultará en un ticket similar al expuesto en la figura 7.41

SHSH-30 Cambiar credenciales personales

La web SPA contendrá un apartado destinado al usuario de cada empleado, desde el cual podrán modificar la contraseña a emplear durante la identificación en el sistema. Esta operación será solo completada si se indica correctamente la contraseña actual e introduciendo por partida doble la contraseña nueva deseada, evitando posibles errores al escribirla. Esta vista se muestra en la figura 7.42.

Figura 7.41: Ejemplo de ticket de venta

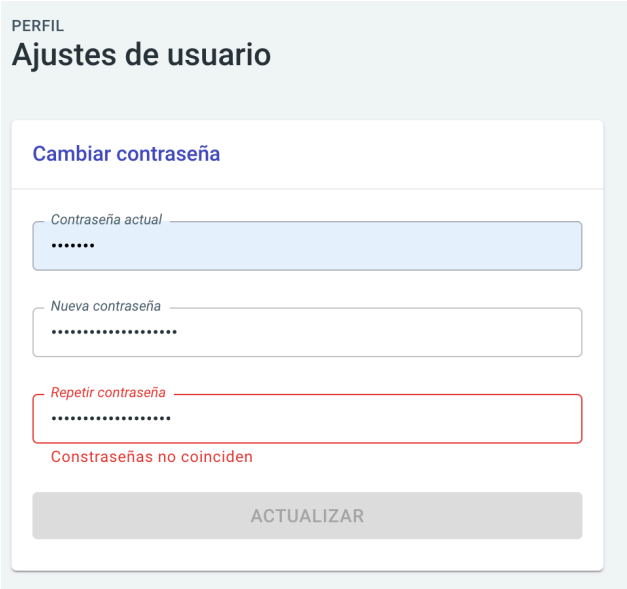


Figura 7.42: Vista del perfil personal

Capítulo 8

Conclusiones y trabajo futuro

8.1 Resultados

Se considera que los objetivos del proyecto fueron cumplidos, diseñando e implementando un sistema funcional y adaptado al modelo de negocio concreto de la empresa destino. Esta actividad permitió demostrar y aplicar las competencias que se obtuvieron a lo largo de la carrera en las materias cursadas.

Con la realización de este proyecto no solo se pretendía dar uso de lo conocido, sino que se quería incorporar nuevos conocimientos que serían útiles en proyectos futuros, destacando:

- Uso de tecnologías en auge para el **desarrollo multiplataforma**, como es Electron.
- Introducción a las **bases de datos documentales con peticiones asíncronas**.
- **Integración con dispositivos externos**, como la incorporación al sistema de una impresora térmica Epson utilizando el SDK proporcionado para desarrolladores.
- Empleo de **herramientas para la gestión de proyectos en metodologías ágiles**.
- Gestión y mantenimiento de entornos para desarrollo con **herramientas de integración e inspección**.
- **Despliegue de aplicaciones en entornos de producción**.

8.2 Relación con la titulación

Entre las materias cursadas durante la titulación destacaremos aquellas que han tenido gran repercusión en el desarrollo del proyecto:

- **Internet y Sistemas Distribuidos**: Esta asignatura nos introdujo en el desarrollo de aplicaciones empresariales, con la implementación de Servicios Web REST, pudiendo

aplicar los conocimientos obtenidos ya que se implementó un servicio que usaba esta arquitectura.

- **Programación Avanzada**: En esta asignatura se continuó con los conocimientos obtenidos en Internet y Sistemas Distribuidos a través del uso de frameworks modernos que facilitan el desarrollo. En esta materia se dieron tecnologías como Spring, React y Redux; cuyo conocimiento pudo aplicarse en la implementación del servicio y las diferentes interfaces.

- **Redes**: Dado que toda la arquitectura de la aplicación a desarrollar se basaba en la comunicación entre distintas máquinas hay que tener siempre en consideración la base obtenida sobre la comunicación entre estas y el uso de los protocolos de comunicación empleados.

- **Herramientas de Desarrollo**: Esta asignatura nos introdujo en el uso de diferentes herramientas útiles de apoyo al proceso de desarrollo software, como los son las herramientas de gestión de proyectos, de control de código, de empaquetado, de integración e inspección continua, etc. Algunas de las herramientas vistas fueron incorporadas para otorgar de mayor calidad al código.

- **Bases de Datos y Bases de Datos Avanzadas**: En el proyecto nos enfrentamos al uso de un gestor de Bases de Datos relacional conocido, MySQL, y con él podremos emplear todo lo aprendido en estas materias.

- **Metodologías de Desarrollo**: Gracias a esta asignatura se pudo adaptar fácilmente la metodología impartida, Scrum.

- **Ingeniería de Requisitos**: Los conocimientos sobre obtención y análisis de requisitos estuvieron presentes en la definición de las historias de usuario.

- **Diseño Software**: Durante el diseño de los Sprint se dio uso de lo aprendido en esta materia, empleando los diagramas de clase y los diagramas de secuencia que marcarían el desarrollo.

- **Interfaces Hombre Máquina**: Se tomó en consideración las lecciones basadas en interfaces responsivas y adaptativas durante la implementación de la interfaz de usuario de la aplicación.

- **Arquitectura Software**: Para el análisis inicial del proyecto los conocimientos obtenidos en esta asignatura fueron muy útiles, ya que nos permitió representar fácilmente la estructura que tendría la solución propuesta.

8.3 Trabajo futuro

La funcionalidad que ofrece el sistema se pretende ampliar con varios apartados que se considera que aportarán bastante valor a las empresas de este modelo de negocio:

- **Gestión de stock**: Es una utilidad bastante característica en los gestores e-commerce y que se desea incorporar al sistema para facilitar y mantener registros sobre esta información. Además se podría incluir el control sobre los lotes de productos adquiridos, con lo que se obtendría un calculo de los beneficios más aproximado.
- **Gestión de eventos**: El objetivo es registrar los eventos de los que se tenga constancia incluyendo detalles importantes como sus fechas de ejecución, localización y costes. Esta información sería útil para notificar sobre próximos eventos, ofrecer un calendario con el plan de actuación, comparación de eventos, etc.
- **Reservas en PoS**: Las reservas en el punto de venta permitiría registrar pedidos o encargos que realice la clientela, además del importe cobrado por adelantado.

Lista de acrónimos

API Application Programming Interface. 32

CLI Command line interface. 34, 38, 39

DAO Data Access Object. 68

IDE Integrated Development Environment. 35

PoS Point of Sale. 3, 12–14, 37, 39, 43, 54–57, 60, 62, 64, 75

REST Representational State Transfer. 2, 3, 30, 37

SCM Source Code Management System. 34

SDK Software Development Kit. 3, 70, 71

SPA Single-Page Application. 3, 16, 32, 37–40, 43, 44, 52, 68, 71

Glosario

e-commerce Un sistema de compra y venta de productos y servicios que utiliza Internet como medio principal de intercambio. 5, 75

framework Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. 2, 3, 29–32, 35, 38, 45, 50, 74

función hash Son funciones que, utilizando un algoritmo matemático, transforman un conjunto de datos en un código alfanumérico con una longitud fija. 47

merchandising Conjunto de productos publicitarios para promocionar un artista, un grupo, una marca, etc. 2

mockup Los mockups son representaciones de media-alta fidelidad que incluyen o simulan la interacción de una interfaz de usuario. En esta representación los usuarios podrán experimentar en alguna medida la experiencia de uso del producto. 9–15

NoSQL Las bases de datos NoSQL están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Las bases de datos NoSQL son ampliamente reconocidas porque son fáciles de desarrollar, por su funcionalidad y el rendimiento a escala. 29, 32

script Un script es una lista de comandos que ejecuta un determinado programa o motor de script. Se pueden utilizar scripts para automatizar procesos en una máquina. 30, 32, 40, 51

Bibliografía

[1] "Web de Shopify." [En línea]. Disponible en: https://www.shopify.es/

[2] "Web de Shopify E-Commerce." [En línea]. Disponible en: https://www.shopify.es/online

[3] "Web de Shopify POS." [En línea]. Disponible en: https://www.shopify.es/pos

[4] "Web de Square." [En línea]. Disponible en: https://squareup.com/us/es

[5] "Web de Ecwid." [En línea]. Disponible en: https://www.ecwid.com/es/

[6] "Web de Vend." [En línea]. Disponible en: https://www.vendhq.com/

[7] K. S. y Jeff Sutherland, "La Guía de Scrum™." 2017. [En línea]. Disponible en: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-European.pdf

[8] "Página web del proyecto Spring." [En línea]. Disponible en: https://spring.io/

[9] "Página web de Spring Boot." [En línea]. Disponible en: https://spring.io/projects/spring-boot

[10] "Página web de Spring Security." [En línea]. Disponible en: https://spring.io/projects/spring-security

[11] "Página web de Spring Data JPA." [En línea]. Disponible en: https://spring.io/projects/spring-data-jpa

[12] "Página web de Apache Maven." [En línea]. Disponible en: https://maven.apache.org/

[13] "Página web de MySQL." [En línea]. Disponible en: https://www.mysql.com/

[14] "Página web de Flyway." [En línea]. Disponible en: https://flywaydb.org/

[15] "Página web de JUnit." [En línea]. Disponible en: https://junit.org/junit5/

[16] "Página web de React." [En línea]. Disponible en: https://es.reactjs.org/

[17] "Página web de Redux." [En línea]. Disponible en: https://es.redux.js.org/

[18] "Página web de Material-UI." [En línea]. Disponible en: https://material-ui.com/es/

[19] "Página web de Material Design." [En línea]. Disponible en: https://material.io/design/

[20] "Herramienta de color de Material Design." [En línea]. Disponible en: https://material.io/resources/color/

[21] "Página web de Jest." [En línea]. Disponible en: https://jestjs.io/

[22] "Página web de Snapshot testing con Jest." [En línea]. Disponible en: https://jestjs.io/docs/en/snapshot-testing/

[23] "Página web de NPM." [En línea]. Disponible en: https://www.npmjs.com/

[24] "Página web de ElectronJS." [En línea]. Disponible en: https://www.electronjs.org/

[25] "Página web de PouchDB." [En línea]. Disponible en: https://pouchdb.com/

[26] "API de PouchDB." [En línea]. Disponible en: https://pouchdb.com/api.html

[27] "Página web de Electron Forge." [En línea]. Disponible en: https://www.electronforge.io/

[28] "Página web de Git." [En línea]. Disponible en: https://git-scm.com/

[29] "Página web de GitHub." [En línea]. Disponible en: https://github.com/

[30] "Página web de Jenkins." [En línea]. Disponible en: https://www.jenkins.io/

[31] "Página web de SonarQube." [En línea]. Disponible en: https://www.sonarqube.org/

[32] "Página web de Jira." [En línea]. Disponible en: https://www.atlassian.com/es/software/jira/

[33] "Página web de IntellJ IDEA." [En línea]. Disponible en: https://www.jetbrains.com/es-es/idea/

[34] "Página web de Balsamiq." [En línea]. Disponible en: https://balsamiq.com/

[35] "Herramienta web Draw.io." [En línea]. Disponible en: https://www.draw.io/

[36] "Página web de StarUML." [En línea]. Disponible en: http://staruml.io/

[37] "Probando la capa Web con Spring." [En línea]. Disponible en: https://spring.io/guides/gs/testing-web/

[38] "Página web de la herramienta create-react-app." [En línea]. Disponible en: https://create-react-app.dev/

[39] "Personalizar Material-UI con un tema." [En línea]. Disponible en: https://material-ui.com/es/customization/theming

[40] "Plantilla de aplicación Electron con Webpack." [En línea]. Disponible en: https://www.electronforge.io/templates/webpack-template

[41] J. Isaacks, *Get Programming with JavaScript Next*, 1st ed. Manning Publications, 2018.

[42] "Guía de instalación de Jenkins." [En línea]. Disponible en: https://www.jenkins.io/doc/book/installing/

[43] "Guía de instalación de SonarQube." [En línea]. Disponible en: https://docs.sonarqube.org/latest/setup/overview/

[44] "Configurar servidor de tareas en IntelliJ." [En línea]. Disponible en: https://www.jetbrains.com/help/idea/tutorial-configuring-generic-task-server.html

[45] "Conectar Jira y cuenta de GitHub." [En línea]. Disponible en: https://confluence.atlassian.com/adminjiracloud/connect-jira-cloud-to-github-814188429.html

[46] "Guía de Smart Commits Jira." [En línea]. Disponible en: https://confluence.atlassian.com/fisheye/using-smart-commits-960155400.html

[47] "Web de notistack." [En línea]. Disponible en: https://iamhosseindhv.com/notistack

[48] "Colección de ilustraciones unDraw." [En línea]. Disponible en: https://undraw.co/illustrations

[49] "Despliegue de aplicación Sprint Boot en linux." [En línea]. Disponible en: https://docs.spring.io/spring-boot/docs/current/reference/html/deployment.html#deployment-service

[50] "Web de NGINX." [En línea]. Disponible en: https://nginx.org/en/

[51] "División de código basado en rutas en React." [En línea]. Disponible en: https://es.reactjs.org/docs/code-splitting.html#route-based-code-splitting

[52] "Librería bcryptjs." [En línea]. Disponible en: https://www.npmjs.com/package/bcryptjs

[53] "Librería uuid." [En línea]. Disponible en: https://www.npmjs.com/package/uuid

[54] "Web de Material-UI Pickers." [En línea]. Disponible en: https://material-ui-pickers.dev/

[55] "Web de Spring Data - Proyecciones." [En línea]. Disponible en: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#projections